НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)
“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему *Програмне застосування для виявлення затемнених зон на рентген знімку легень*

Виконав: студент IV курсу, групи *ІП-61 Данилюк Микола Іванович* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *старший викладач, к. т. н., Сирота О. П.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Консультант з графічної документації *доц., к.т.н., Ліщук К.І.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Рецензент:

аспірант, Дьяков Сергій Олександрович
посада,науковий ступінь,вчене звання,прізвище,ім’я, по батькові (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

Дата перевірки:
16.06.2020 03:35:09 EEST

Дата звіту:
16.06.2020 03:59:54 EEST

ID перевірки:
1004063791

Тип перевірки:
Doc vs Internet + Library

ID користувача:
77149

Назва документу: Danyliuk_ip61_2

ID файлу: 1004076762 Кількість сторінок: 34 Кількість слів: 5769 Кількість символів: 41454 Розмір файлу: 137.35 KB

15.9% Схожість

Найбільша схожість: 8.6% з джерело бібліотеки. ID файлу: 1000020588

6.41% Схожість з Інтернет джерелами	208	Page 36
14.2% Текстові збіги по Бібліотеці акаунту	374	Page 37

0.24% Цитат

Цитати	1	Page 38

Вилучення переліку посилань вимкнено

0% Вилучень

Вилучений текст відсутній

Підміна символів

Заміна символів	1

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних управляючих систем та технологій*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис)
“___”_____2020 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Данилюк Миколи Івановича
(прізвище, ім’я, по батькові)

1. Тема проєкту « *Програмне застосування для виявлення затемнених зон на рентген знімку легень*»

керівник проєкту *Сирота Олена Петрівна, к.т.н., старший викладач*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “07” травня 2020 р. №1081-с

2. Термін подання студентом проєкту «08» *червня 2020 року*

3. Вихідні дані до проєкту

Технічне завдання

4. Зміст пояснювальної записки

1) Аналіз вимог до програмного забезпечення: загальні положення, змістовний опис аналіз предметної області, аналіз відомих рішень,аналіз вимог до програмного забезпечення

2) Інформаційне забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, конструювання програмного забезпечення,аналіз безпеки даних

3) Аналіз якості та тестування програмного забезпечення: аналіз якості ПЗ, опис процесів тестування, аналіз якості тренування моделі

4) Впровадження та супровід програмного забезпечення

5. Перелік графічного матеріалу

1) Структура мережі

2) Схема структурна класів програмного забезпечення

3) Схема структурна розгортання

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» *березня 2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*10.04.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*10.04.2020*	
3.	*Постановка та формалізація задачі*	*17.04.2020*	
4.	*Аналіз вимог до програмного забезпечення*	*17.04.2020*	
5.	*Моделювання програмного забезпечення*	*24.04.2020*	
6.	*Обґрунтування використовуваних технічних засобів*	*24.04.2020*	
7.	*Розробка архітектури програмного забезпечення*	*24.04.2020*	
8.	*Розробка програмного забезпечення*	*01.05.2020*	
9.	*Налагодження програми*	*08.05.2020*	
10.	*Виконання графічних документів*	*15.05.2020*	
11.	*Оформлення пояснювальної записки*	*15.05.2020*	
12.	*Подання ДП на попередній захист*	*15.05.2020*	
13.	*Подання ДП рецензенту*	*01.06.2020*	
14.	*Подання ДП на основний захист*	*10.06.2020*	

Студент _____ **Микола ДАНИЛЮК**
(підпис)

Керівник _____ **Олена СИРОТА**
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	КПІ.ІП-6107.045490.01.90	Відомість проекту	1	
3	А4	КПІ.ІП-6107.045490.02.81	Пояснювальна записка	50	
4	А4	КПІ.ІП-6107.045490.03.91	Технічне завдання	11	
5	А4	КПІ.ІП-6107.045490.04.13	Опис програми	8	
6	А4	КПІ.ІП-6107.045490.05.34	Керівництво користувача	3	
7	А3	КПІ.ІП-6107.045490.06.99 Р	Структура генеративної мережі	1	
8	А3	КПІ.ІП-6107.045490.06.99 СС	Схема структурна класів програмного забезпечення	1	
9	А3	КПІ.ІП-6107.045490.06.99 СС	Схема структурна розгортання	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-6107.045490.01.90		
Розробн.		Данилюк М. І.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Сирота О. П.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-61	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проєкту

на тему: Програмне застосування для виявлення затемнених зон на рентген знімку легень

Київ – 2020 року

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з 4 розділів, містить 83 сторінки, 28 рисунків, 19 таблиць, 2 додатки та 11 джерел.

Дипломний проект присвячено розробці утиліти визначення зон затемнення на рентген знімку грудної клітки.

Розробка може використовуватись для інтеграції з існуючими системами медичних ПЗ.

Розділ інформаційне забезпечення містить інформацію про вхідні та вихідні дані, було описано звідки формується навчаюча вибірка для згорткової нейронної мережі та описана структура зберігання даних, необхідних для роботи системи.

Розділ математичного забезпечення присвячений обґрунтуванню обраного підходу вирішення поставленої задачі та опису його роботи.

У розділі програмного та технічного забезпечення описано мінімальні вимоги до забезпечення для успішного використання програмного продукту.

У технологічному розділі описується інструкція користувача та тестування роботи системи.

КОМП’ЮТЕРНИЙ ЗІР, МАШИННЕ НАВЧАННЯ. НЕЙРОННІ МЕРЕЖІ, ВИЯВЛЕННЯ ОБ’ЄКТІВ, МЕДИЧНЕ ПЗ

ABSTRACT

Diploma project consists of 5 sections, contains 28 drawings, 19 tables, 2 applications, 11 sources.

The diploma project is devoted to the development of Software for Lung Opacities Detection on X-ray Picture.

The development can be used to integrate with existing medical software systems.

The information support section describes the input and output data to the system, describes where is the training set is and describes the storage structure necessary for the system to work.

The section of mathematical support is devoted to the substantiation of the chosen approach to the solution and the description of its work.

The software describes system development tools, hardware requirements and software architectures.

The technology section describes the user's manual and tests a set of tasks.

COMPUTER VISION, MACHINE LEARNING, NEURAL NETWORKS, OBJECT DETECTION, MEDICAL SOFTWARE

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CXR – chest X-ray, рентген знімок грудної клітки.

RSNA – Radiological Society of North America.

Бібліотека – збірник програм, що виконують деякі типові задачі та застосовуються для полегшення розробки програмного забезпечення.

CNN – convolutional neural network, згорткова нейронна мережа.

Машинне навчання – вивчення алгоритмів та математичних моделей, які комп'ютерні системи використовують для ітеративного покращення їх продуктивності для конкретної задачі.

Комп'ютерний зір – область, що вирішує проблему створення комп'ютерів, що можуть отримувати знання з цифрових картинок чи відео подібно до того, як це робить людина.

IoU – Intersection Over Union

ROI – Region of Interest

ВСТУП

Метою створення розробки є написання алгоритму машинного навчання з виявлення затемнених зон на рентген знімку грудної клітки. Наявність цих елементів на CXR є найвірогіднішою ознакою існування у пацієнта такої хвороби, як пневмонія.

Ось передісторія та чому важливо вирішити цю проблему.

Пневмонія становить понад 15% усіх смертей дітей до 5 років у світі. У 2015 році від захворювання померло 920 000 дітей віком до 5 років. В Україні у 2018 році смертей від пневмонії було 2,5 тисячі. У 2019 на 5% більше.

Актуальність теми не визиває сумнівів.

Хоча часто, точно діагностувати пневмонію - це важке завдання. Це вимагає огляду рентген знімку грудної клітки висококваліфікованими фахівцями та підтвердження через клінічну історію, життєві ознаки та лабораторні обстеження. Пневмонія зазвичай проявляється як область або ділянки затемненої зони на CXR. Однак діагностика пневмонії на CXP є складною через низку інших станів у легенях, таких як перевантаження рідини (набряк легенів), кровотечі, втрата об’єму (ателектаз або колапс), рак легенів або післяпроменеві або хірургічні зміни. Поза легенів рідина в плевральному просторі (плевральний випіт) також виявляється як підвищена непрозорість на CXR. За наявності, порівняння КХР пацієнта, прийнятого в різні моменти часу, та співвідношення з клінічними симптомами та анамнезом є корисними для постановки діагнозу.

CXR - це найчастіше проведене діагностичне візуалізація. Ряд факторів, таких як позиціонування пацієнта та глибина вдиху, можуть змінити появу CXR, ускладнюючи інтерпретацію. Крім того, клініцисти стикаються з читанням великих обсягів зображень у кожну зміну.

Для підвищення ефективності та охоплення діагностичних послуг Радіологічне товариство Північної Америки співпрацювало разом з

Національним інститутом здоров’я США та MD.ai для розробки великого набору даних для машинного навчання.

Мета розробки полягає у побудові моделі машинного навчання, яка зможе передбачати по рентген знімку грудної клітки передбачати зони затемнення. Потім результат цієї моделі треба обгорнути у REST сервіс, який буде зручно інтегрувати в існуючі системи медичного забезпечення. Фінальний кроком буде написання веб додатку, за допомогою якого можна буде наочно продемонструвати роботу REST сервісу.

Практичне значення полягає у автоматизації огляду рентген знімків грудної клітки пацієнтів, яким прогнозують пневмонію. З огляду на велику завантаженість відповідних спеціалістів, це принесе вклад у розв’язання проблеми.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Одним з найбільш прогресивних напрямків розвитку інформаційних технологій є машинне навчання. Машинне навчання (англ. Machine learning) – це підгалузь штучного інтелекту в галузі інформатики, яка часто застосовує статистичні прийоми для надання комп'ютерам здатності «навчатися» (тобто, поступово покращувати продуктивність у певній задачі) з даних без явного програмування. Еволюціонувавши з досліджень розпізнавання образів та теорії обчислювального навчання в галузі штучного інтелекту, машинне навчання досліджує вивчення та побудову алгоритмів, які можуть навчатися й робити прогнозування, опираючись на наявні дані – такі алгоритми долають слідування строго статичним програмним інструкціям, здійснюючи керовані даними прогнози або ухвалювання рішень шляхом побудови моделі з вибіркових входів. Машинне навчання застосовують у ряді обчислювальних задач, у яких розробка та програмування явних алгоритмів з доброю продуктивністю є складною або нездійсненною; до прикладів таких додатків належать фільтрування електронної пошти, виявляння мережних вторгників або зловмисних інсайдерів, що добиваються витоку даних, оптичне розпізнавання символів (ОРС), навчання ранжуванню та комп'ютерний зір [1].

За останні кілька десятиліть дослідження комп’ютерного зору, обробки зображень та розпізнавання візерунків досягли значного прогресу. Крім того, медична візуалізація привертає все більше уваги в останні роки завдяки її життєво важливим компонентом у галузі охорони здоров'я. Дослідники опублікували безліч базових наук та даних, що підтверджують прогрес та застосування охорони здоров'я в галузі медичних зображень. На Рис. 1.1 зображено сегментацію органів тіла.

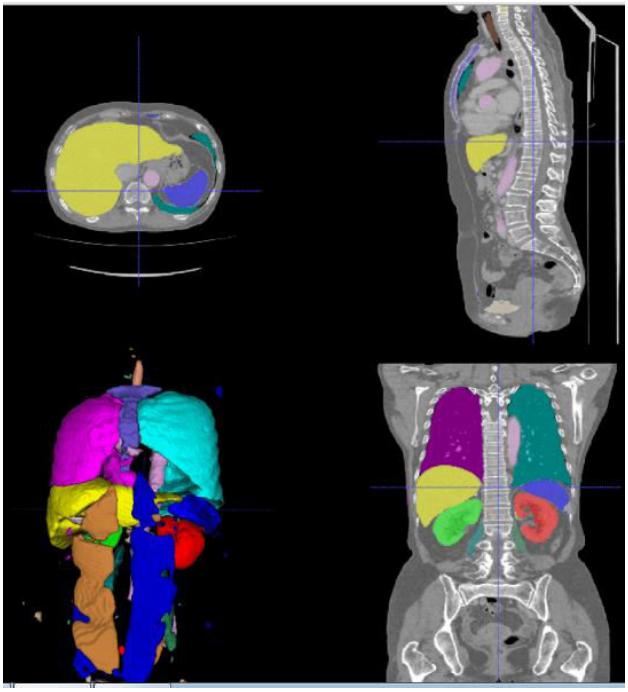


Рисунок 1.1 - Сегментація органів тіла

1.2 Змістовний опис і аналіз предметної області

Рисунок 1.2 - Анатомія грудної клітки

На Рис 1.2 зображена анатомія грудної клітки з виділеними легенями.

Ви можете бачити, що є маса тканини, що оточує легені та між легенями. Ці області містять шкіру, м’язи, жир, кістки, а також серце та великі судини.

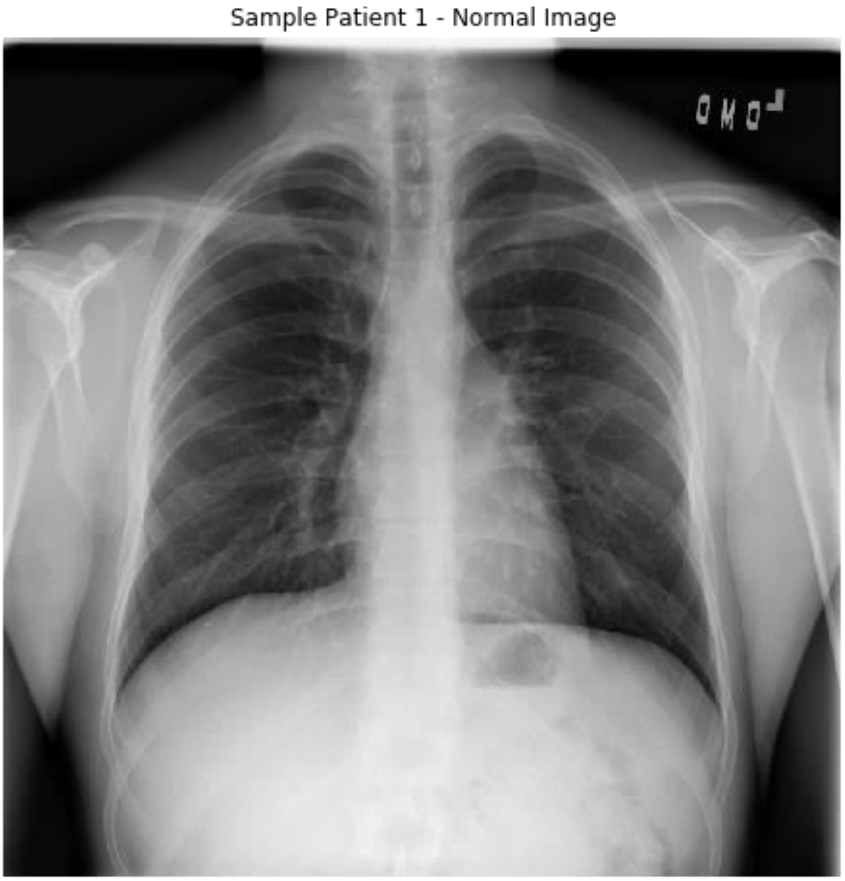


Рисунок 1.3 – Рентген знімок здорових легень

Це приклад нормальної рентгенографії грудної клітки (CXR) з гарною технічною якістю.

Розкажу трохи про основи рентгенографії грудної клітки.

У процесі зйомки зображення рентген проходить через тіло і доходить до детектора з іншого боку. Тканини з розрідженим матеріалом, наприклад легені, наповнені повітрям, не поглинають рентгенівські промені і на зображенні виглядають чорними. Щільні тканини, такі як кістки, поглинають рентгенівські промені і на зображенні здаються білими.

Грубо кажучи, відповідність наступна:

- чорне – повітря;
- біле – кістка;

- сіре - тканина або рідина.

Ліва частина об'єкта за умовами знаходиться в правій частині екрана. Ви також можете побачити малу L у верхньому правому куті. На звичайному зображенні легені ми бачимо як чорні, але на них різні проекції - в основному кістки грудної клітки, головні дихальні шляхи, кровоносні судини та серце.

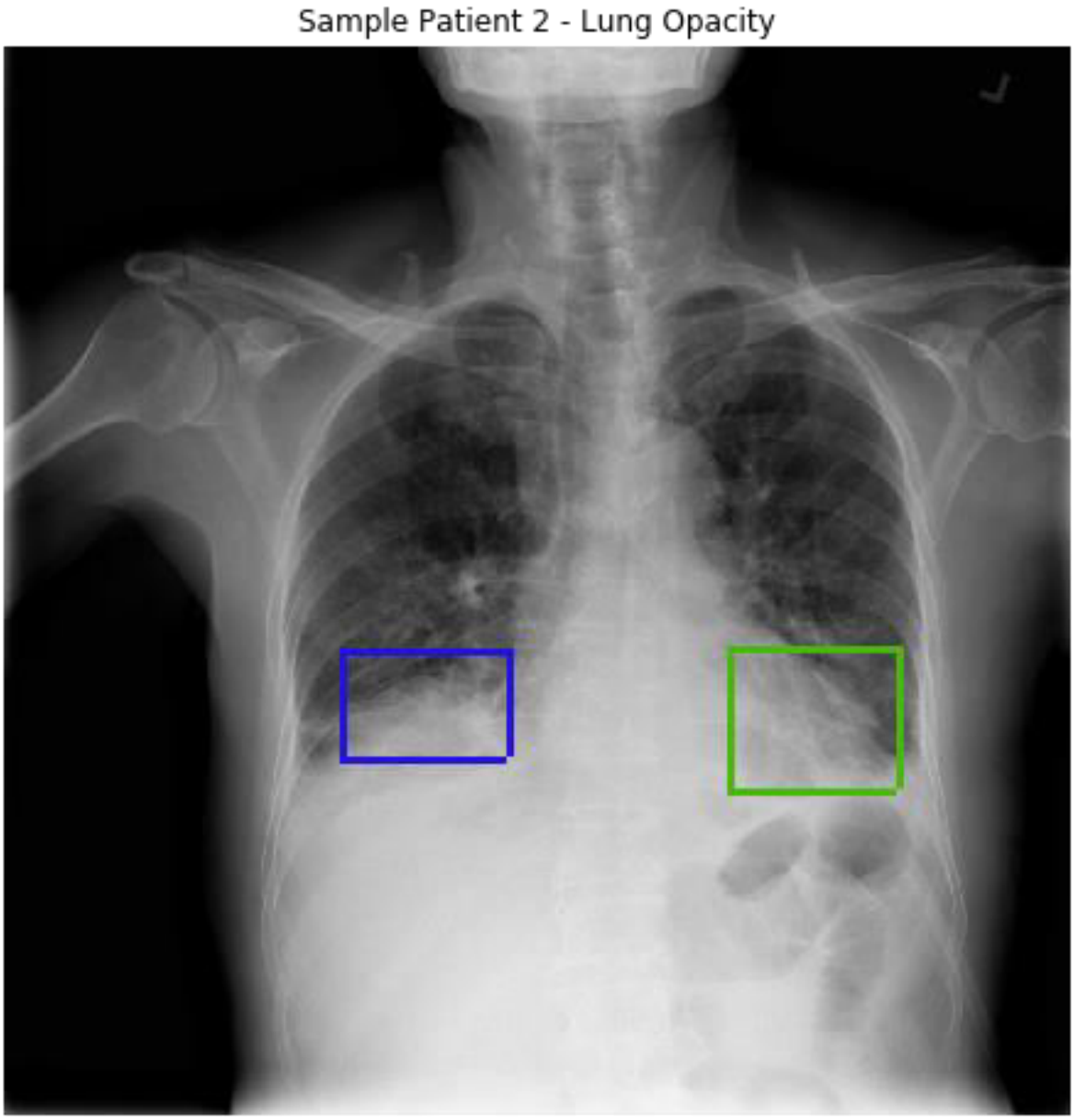


Рисунок 1.4 - Рентген знімок хворих легень

Що таке зона затемнення легенів?

Зона затемнення(lung opacity) позначає будь-яку зону, яка переважно зменшує рентгенівський промінь і тому виявляється більш непрозорою, ніж

навколишня область. Це неспецифічний термін, який не вказує на розмір чи патологічний характер аномалії [2].

Значення(meaning) - будь-яка ділянка на рентгенограмі грудної клітки, яка є білішою, ніж повинна бути. Якщо порівняти Рисунок 1.3 та Рисунок 1.4, то можна побачити, що нижня межа легенів хворого пацієнта затьмарена непрозорістю. На зображенні зразка здорового пацієнта видно чітку різницю між чорними легенями і тканиною під ним, а на Рисунку 1.4 є саме ця нечіткість.

Зазвичай легені переповнені повітрям. Коли у когось є пневмонія, повітря в легенях замінюється іншим матеріалом - рідинами, бактеріями, клітинами імунної системи і т. Д. Ось чому ділянки помутніння - це ділянки сірого кольору, але повинні бути більш чорними. Коли ми бачимо їх, ми розуміємо, що легенева тканина в цій області, ймовірно, не здорова.

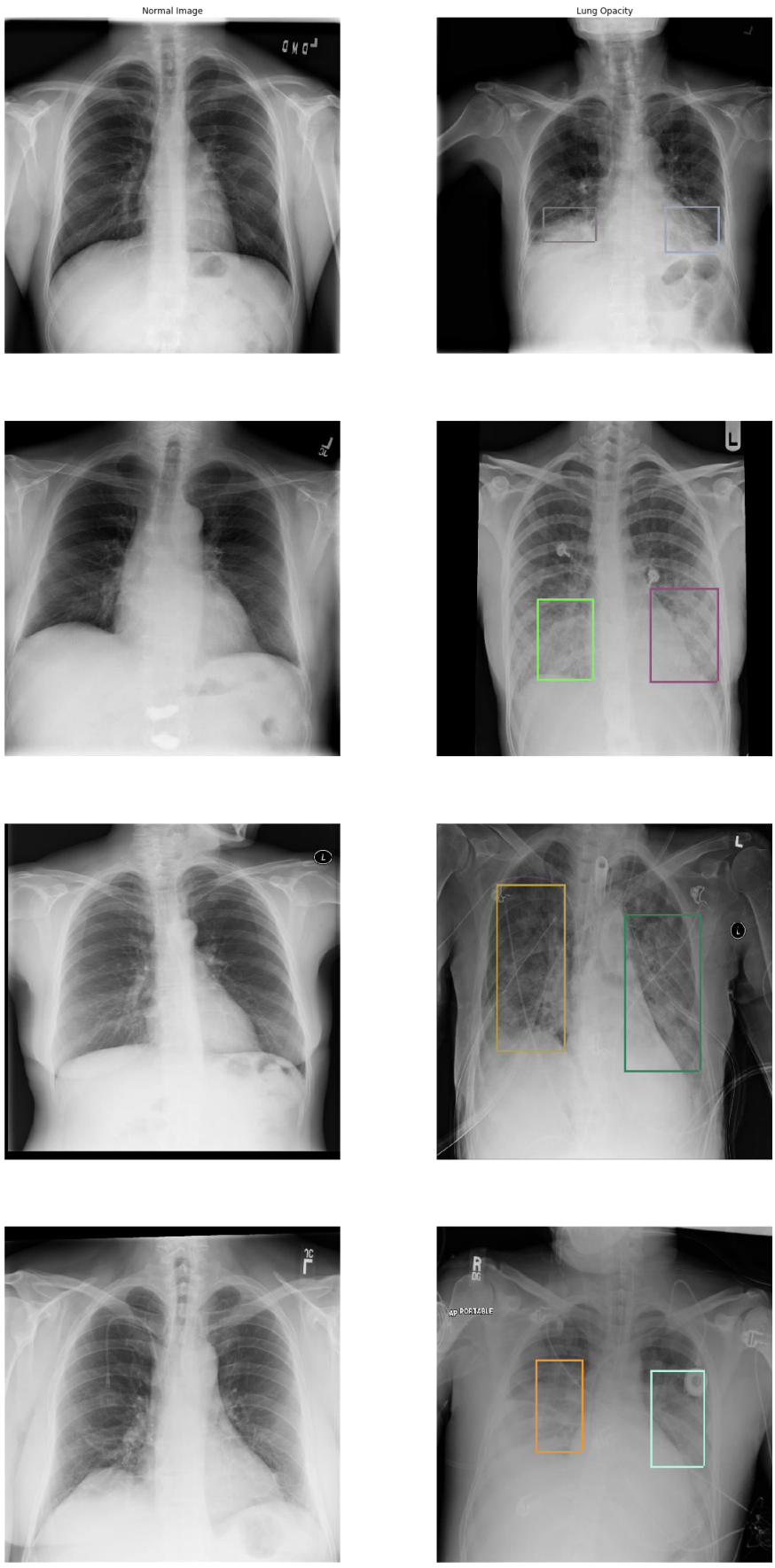


Рисунок 1.5 - Приклади рентген знімків хворих легень

На зображеннях непрозорості легенів ми можемо побачити, що існує невідомість, якщо мічені ящики (це називається непрозорістю земного скла) та / або втрата звичайних меж легенів (називається консолідацією).. Ви також можете бачити, що хворі на пневмонію хворіють і мають різні кабелі, наклейки та трубки, підключені до них. Якщо ви бачите круглу білу малу непрозорість в легенях і навколо неї, ймовірно, наклейка на ЕКГ.

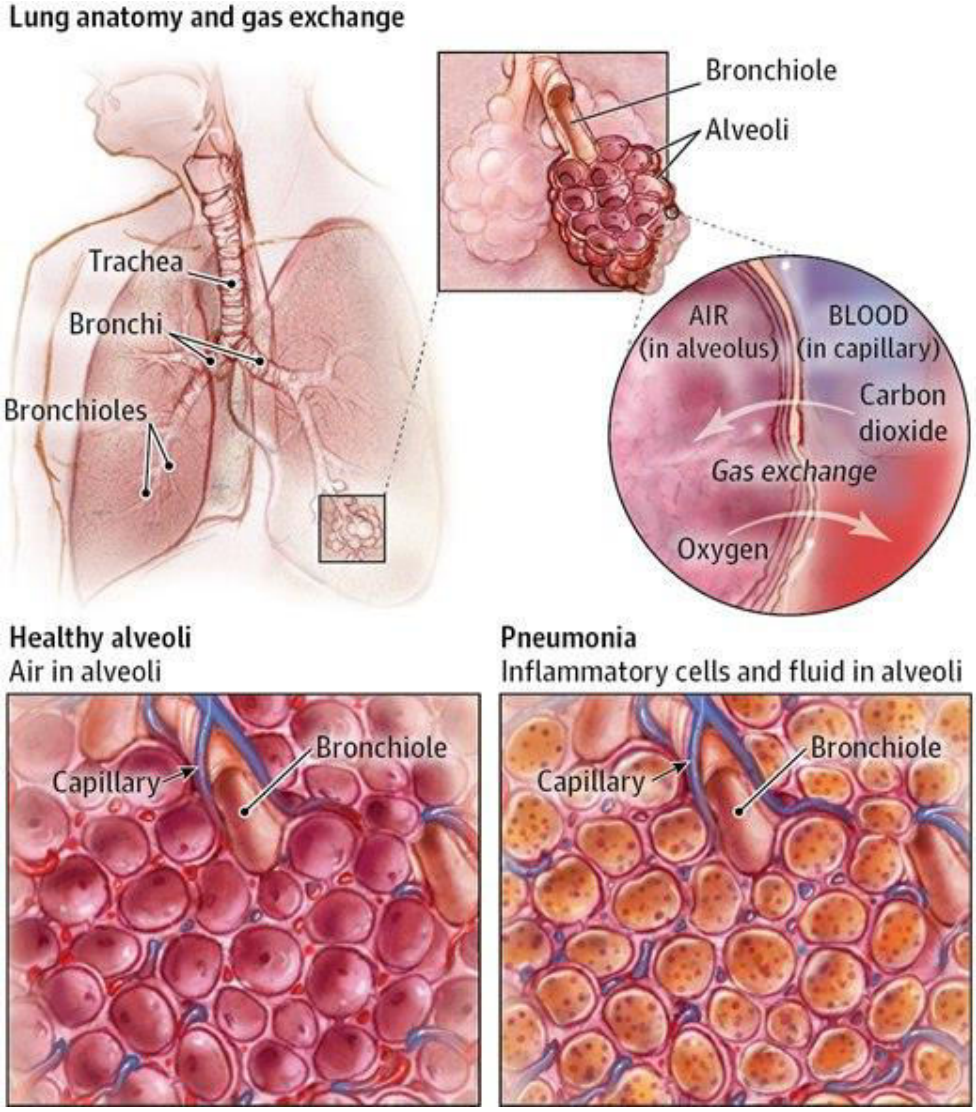


Рисунок 1.6 - Анатомія легень

Пневмонія - це легенева інфекція, яка може бути викликана бактеріями, вірусами або грибками. Через інфекцію та імунну відповідь організму мішки в легенях (їх називають альвеолами) наповнюються рідиною замість повітря.

Причиною того, що пневмонія, пов'язана з помутнінням легенів, дифузно виглядає на рентгенограмі грудної клітки, тому що інфекція та рідина, що накопичуються, поширюються в межах нормального дерева дихальних шляхів у легенях. Немає чіткої межі, де інфекція припиняється. Це відрізняється від інших захворювань, таких як пухлини, які абсолютно відрізняються від нормальних легенів і не підтримують нормальну структуру дихальних шляхів всередині легені [3].

1.3 Аналіз відомих досліджень

1.3.1 CheXNet, Standford Univercity

Під час пошуку схожих рішень натрапив на наукову статтю вчених з Стенфордського Університету «CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning» [4].

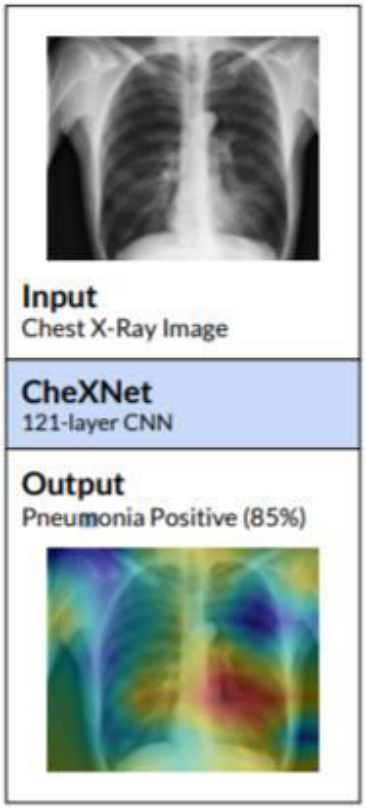


Рисунок 1.7 - CheXNet

CheXNet (Рисунок 1.7), являє собою 121-рівневу згорткову нейронну мережу. Вхідними даними є рентген знімок грудної клітки, вихідні дані - ймовірність виникнення пневмонії разом з тепловою картою, що локалізує ділянки зображення, найбільш показані для пневмонії.

Проводилось навчання CheXNet за набором даних ChestX-ray14 [5] який містить 112,120 рентгенівських знімків грудної клітки, окремо позначених до 14 різних захворювань грудної клітки, включаючи пневмонію.

Ваги мережі ініціалізуються вагами з моделі, перевіреної на ImageNet. Мережа тренується, використовуючи оптимізатор Adam зі стандартними параметрами ($\beta1 = 0,9$ і $\beta2 = 0,999$). Модель тренується за допомогою батчів розміром 16. Початковий learning rate = 0,001, що зменшується в 10 разів щоразу, коли після епохи validation loss виходить на плато, і вибираємо модель з найменшими втратами на перевірку.

1.3.2 RetinaNet Pneumonia Detector, Dmytro Poplavskiy

Розв’язок цього дослідника заснована на модифікованій моделі на основі RetinaNet. Одинарна модель, зібрані виходи в 4 рази.[6]

Для аугментації використав легкі обертання (до 6 град.), Зсув, масштаб, зсув і горизонтальне обернення, для деяких зображень випадковий рівень розмиття та шуму та зміни гами. Обмежив кількість яскравості / гамма-збільшення, тому що мені було важко перевірити, чи не недійсні вони мітки. Щоб зменшити вплив обертання на розміри обмежувальної коробки, замість обертання кутів обертав дві точки на кожному краю, на 1/3 та 2/3 довжину ребра від кута, загалом 8 балів і обчислював новий обмежувальний ящик як min / max повернутих точок.

Тренував модель близько 12 епох, приблизно 1 годину за епоху на 1080ti GPU.

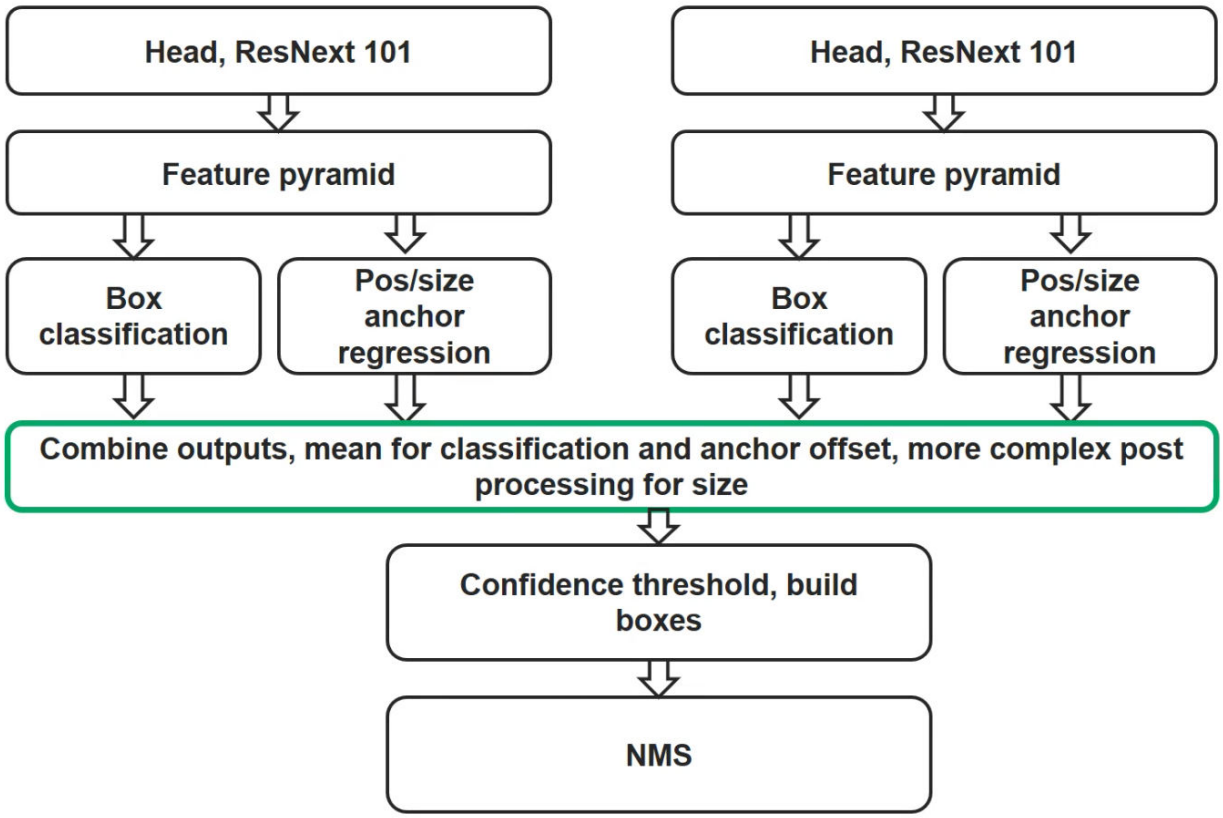


Рисунок 1.8 - Ансамбль моделей

1.3.3 Pneumonia Detection, Ian Pan

Це робота розробника з Brown University, Providence Ian Pan, який проводив навчання своєї моделі на тому ж датасеті від RSNA [7]. У свої роботі він використав ансамбль з 5 10-fold моделей для тренування. В якості алгоритмів використовував RetinaNet, Deformable R-CNN, Deformable Relation Networks.

1.4 Аналіз вимог до програмного забезпечення

Виходячи з аналізу наявного функціоналу основних конкурентів, їх основних переваг та недоліків можна сформувати ряд функціональних та нефункціональних вимог, які необхідні для створення конкурентного програмного забезпечення. Розроблюване програмне забезпечення також не буде мати функціоналу по авторизації користувачів, оскільки дана інформація не потрібна для повноцінно функціонування веб-додатка з деблюрінгом

зображення та спростить використання розробленого додатку в Європейському союзі, оскільки жодна інформація про користувачі не буде зберігатися на сайті, а отже неможливо ідентифікувати кому належить оброблювана інформація та будь-як використати дану інформацію.

Для цього була поставлена ціль розробити програмний комплекс для тренування та аналізу моделей машинного навчання для розв`язування задачі визначення заметнених зон на легенях та зручний веб-сервіс, для подальшої інтеграції у складнішу медичну систему.

1.4.1 Розроблення функціональних вимог

В системі передбачено наступні функціональні варіанти використання:

Таблиця 1.1 - Варіант використання UC001

Назва	Передбачення зон затемнення легень на рентген знімку грудної клітки
Опис	Пошук обмежувальних ящиків, які відповідають за зони затемнення легень, на зображенні грудної клітки
Учасники	Користувач системи
Передумови	Користувач загрузив на сервер рентген знімок грудної клітки
Післяумови	Користувач отримав відповідь з зображенням грудної клітки та обмежувальними ящиками, які вірогідно позначають зони затемнення легень
Основний сценарій	- Користувач відкриває веб-застосунок. - Користувач натискає кнопку «Choose File» та обирає на своєму ПК файл з зображенням рентген знімка грудної клітки. - Користувач натискає кнопку «Submit». - Система виконує обробку вхідного файлу та відображає зображення з обмежувальними ящиками, які

	вірогідно позначають зони затемнення легень.

Виходячи з вище описаних вимог та аналізу можна сформувати наступні функціональні вимоги:

Таблиця 1.2 - Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Точність моделей
Опис	Моделі машинного навчання в даній роботі повинні мати точність, підраховану як суму average precision з IoU=0.5, не менше 0.1

1.4.2 Розроблення нефункціональних вимог

Програмне забезпечення повинне відповідати наступним нефункціональним вимогам:

Таблиця 1.3 - Опис нефункціональної вимоги REQ002

Номер	REQ002
Назва	Час обробки запиту
Опис	Моделі машинного навчання в даній роботі мають виконувати внутрішні обчислення менш ніж за 3 с

Таблиця 1.4 - Опис нефункціональної вимоги REQ003

Номер	REQ003
Назва	Обробка файлів з неправильним форматом
Опис	Веб-застосунок не має виконувати обробку вхідних даних з неправильним форматом

Таблиця 1.5 - Опис нефункціональної вимоги REQ004

Номер	REQ004
Назва	Локалізація інтерфейсу
Опис	Локалізація інтерфейсу - англійська

Таблиця 1.6 - Опис нефункціональної вимоги REQ005

Номер	REQ005
Назва	Підтримувана версія браузера
Опис	Підтримувана версія браузера: Google Chrome 49 або вище

1.4.3 Постановка комплексу завдань модулю

Розроблюваний програмний продукт призначений для вирішення задачі визначення зон затемнення на рентген знімку грудної клітки.

Мета роботи – створення системи для тренування та аналізу моделей машинного навчання для розв’язування задачі визначення зон затемнення на рентген знімку грудної клітки, також створення зручного веб-сервісу для подальшої інтеграції у складнішу медичну систему.

Розроблений веб-застосунок повинен працювати на пристроях зі встановленим браузером Chrome.

Для реалізації поставленої мети необхідно розв’язати наступні задачі:

- провести аналіз досліджень у сфері розпізнавання затемнених зон на рентген знімку грудної клітки у галузі радіофізики, та використати їхній досвід у підборі конфігурацій для нейронної мережі;
- створити інтерфейс для використання програмного продукту в якому користувач буде мати можливість загрузити вхідне зображення рентген знімка грудної клітки для подальшої обробки та переглянути результат обробки;

- створити архітектуру загорткової нейронної мережі, яка найкраще підходить для поставленої задачі розпізнавання затемнених зон на рентген знімку грудної клітки;
- підібрати гіперпараметри згорткової нейронної мережі для найкращої точності тренування моделі;
- провести аналіз результатів роботи системи на точність визначення та швидкість обробки запиту.

1.5 Висновки по розділу

Отже, у даному розділі був приведений опис предметного середовища.

Було проаналізовано анатомію грудної клітки та легень.

Розглянуто наукові роботи на основі яких зараз розроблено програмне забезпечення, яке використовується в існуючих медичних системах. На їх основі були розроблені основні функціональні та не функціональні вимоги до розроблюваного програмного забезпечення.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

2.1.1 Вхідні дані

У якості вхідних даних використовуються введені користувачем рентген знімки грудної клітки. Вони подаються у форматі DICOM.

Цифрові зображення та комунікації в медицині (DICOM) є стандартом для комунікації та управління медичною візуальною інформацією та суміжних даних.

Також до вхідних даних відноситься файл зі збереженими вагами натренованої на існуючих зображеннях згорткової нейронної мережі, які зберігаються у файлі з назвою «weights.h5». Це необхідно для того, щоб при подальшому використанні системи, вона могла при запуску не тренувати згорткову нейронну мережу з нуля, а використати вже існуючи ваги натренованої мережі з попереднього запуску програми, тим самим заощадивши обчислювальні ресурси комп'ютера та значно зменшивши час запуску програми

Рисунок 2.1 – Вхідне зображення

2.1.2 Вихідні дані

Вихідним документом є зображення грудної клітки з передбаченими зонами затемнення легень. Вони відправляються у форматі PNG.

2.1.3 Структура масивів інформації

Для розробки програмного продукту використовується датасет, який Radiological Society of North America (RSNA®) розробила в колаборації з US National Institutes of Health, The Society of Thoracic Radiology, і MD.ai [8].

Датасет містить 26684 рентген знімки грудної клітки. Кожне зображення надане у форматі DICOM. Також є файл з списком bounding box на кожному зображеннях затемнених зон у наступному форматі: Опис даних у датасеті наведено у Таблиці 2.1.

Таблиця 2.1 - Опис датасету

ID	Опис
patientID	Ідентифікатор рентген знімка пацієнта. Кожному patientId відповідає унікальне зображення з датасету
X	Ліва верхня координата x bounding box
Y	Ліва верхня координата y bounding box
Width	Ширина bounding box
Height	Довжина bounding box

Якщо на зображенні немає bounding box, то у колонках x, y, width, height стоять значення NaN.

Ось як це виглядає:

	patientId	x	y	width	height
0	0004cfab-14fd-4e49-80ba-63a80b6bddd6	NaN	NaN	NaN	NaN
1	00313ee0-9eaa-42f4-b0ab-c148ed3241cd	NaN	NaN	NaN	NaN
2	00322d4d-1c29-4943-afc9-b6754be640eb	NaN	NaN	NaN	NaN
3	003d8fa0-6bf1-40ed-b54c-ac657f8495c5	NaN	NaN	NaN	NaN
4	00436515-870c-4b36-a041-de91049b9ab4	264.0	152.0	213.0	379.0

Рисунок 2.2 - Датасет

2.1.4 Алгоритм

У даній роботі я використав метод Mask R-CNN в зв’язку з його простотою та сучасними характеристиками.

Mask R-CNN - це глибока нейронна мережа, спрямована на вирішення проблеми сегментації екземплярів у машинному навчанні чи комп'ютерному зорі. Іншими словами, вона може відокремлювати різні об’єкти в зображенні чи відео. Ви даєте їй зображення, вона дає вам обмежувальні поля об'єктів, класи та маски.

Існує два етапи Mask R-CNN. По-перше, вона формуються пропозиції про регіони, де може бути об'єкт на основі вхідного зображення. По-друге, прогнозується клас об'єкта, уточнюється обмежувальне поле та формується маска в піксельному рівні об'єкта на основі пропозиції першого етапу. Обидва етапи з'єднані з структурою backbone.

Backbone - це глибока нейронна мережа стилю FPN. Вона складається із шляху "знизу вгору", шляху "зверху вниз" та бічних з'єднань. Шлях знизу вгору може бути будь-яким ConvNet, зазвичай ResNet або VGG, який витягує функції із необроблених зображень. Шлях зверху вниз створює функцію піраміди, яка за розмірами схожа на шлях знизу вгору. Бічні з'єднання - це згортання та додавання операцій між двома відповідними рівнями двох шляхів. FPN перевершує інші одиночні ConvNets в основному з тієї причини, що він підтримує сильні семантичні функції в різних масштабах роздільної здатності.

Тепер розглянемо перший етап. Легка нейрональна мережа під назвою RPN сканує весь шлях FPN зверху вниз (далі - карта функції) і пропонує регіони, які можуть містити об'єкти. Це все. Хоча сканування карти функцій є ефективним способом, нам потрібен метод прив’язання функцій до місця розташування зображення. Ось приходять якоря. Якіри - це набір коробок із заздалегідь визначеними місцями та масштабами щодо зображень. Класи первинної істини (лише об'єкти або фонові бінарні класифіковані на цьому етапі) та обмежувальні поля присвоюються окремим якорям відповідно до деякого значення IoU. Оскільки якорі з різними масштабами прив’язуються до різних рівнів карти об’єктів, RPN використовує ці якорі, щоб визначити, де з карти функції "повинен" отримати об'єкт і який розмір його обмежувального поля. Тут ми можемо погодитись, що згортання, зменшення розміру та перетворення в інший спосіб зберігатимуть функції, що залишаються такими ж відносними, як об’єкти в оригінальному зображенні, і не змішать їх.

На другому етапі інша нейронна мережа приймає запропоновані регіони за першим етапом і присвоює їх декільком конкретним областям рівня функції карти, сканує ці області та генерує об'єкти-класи (багатокатегоріальна класифікація), обмежуючи поля та маски. Процедура виглядає аналогічно RPN. Відмінності полягають у тому, що без допомоги якорів, другий етап використовував трюк під назвою ROIAlign для пошуку відповідних областей карти функцій, і існує гілка, що генерує маски для кожного об’єкта на рівні пікселів.

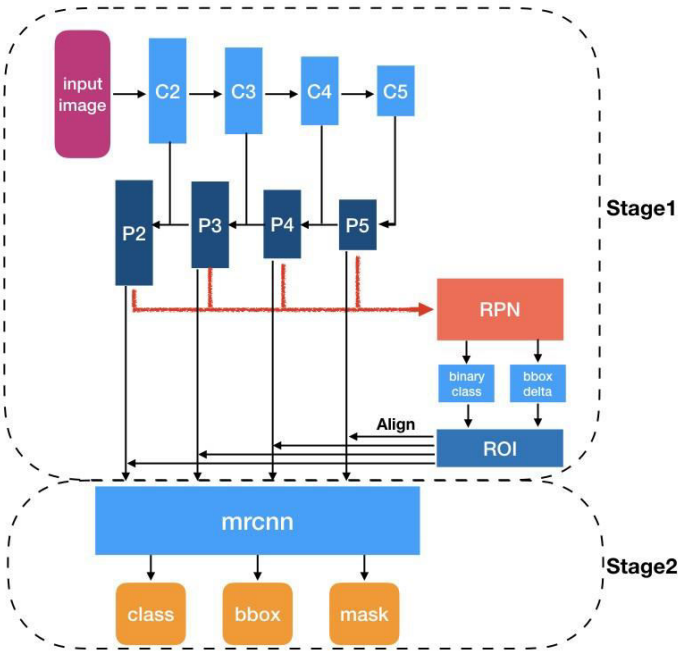


Рисунок 2.3 - Mask R-CNN

2.1.5 Вибір гіперпараметрів

Конфігурація для параметрів детектора Mask R-CNN наведена на Рисунку 2.2.

```
# Train on 1 GPU and 8 images per GPU. We can put multiple images on each
# GPU because the images are small. Batch size is 8 (GPUs * images/GPU).
GPU_COUNT = 1
IMAGES_PER_GPU = 8

BACKBONE = 'resnet50'

NUM_CLASSES = 2  # background + 1 pneumonia classes

IMAGE_MIN_DIM = 256
IMAGE_MAX_DIM = 256
RPN_ANCHOR_SCALES = (16, 32, 64, 128)
TRAIN_ROIS_PER_IMAGE = 32
MAX_GT_INSTANCES = 4
DETECTION_MAX_INSTANCES = 3
DETECTION_MIN_CONFIDENCE = 0.78  ## match target distribution
DETECTION_NMS_THRESHOLD = 0.01

STEPS_PER_EPOCH = 200
```

Рисунок 2.4 - Конфігурація Mask R-CNN

У якості магістральної архітектури я використав ResNet-50.

ResNet-50 – глибока залишкова мережа. 50 означає кількість кількість шарів, які вона має. Це підклас згорткової нейронної мережі, який найчастіше використовується для класифікації зображень. Основним нововведенням ResNet є пропускне з'єднання. Як відомо, без коригування глибокі мережі часто страждають від зникаючих градієнтів, тобто: у міру зворотного поширення моделі градієнт стає все меншим і меншим. Крихітні градієнти можуть зробити навчання непереборним. Пропускне з'єднання на наведеній нижче схемі позначено "ідентичність". Це дозволяє мережі вивчити функцію ідентичності, що дозволяє їй передавати вхід через блок, не проходячи через інші вагові шари.

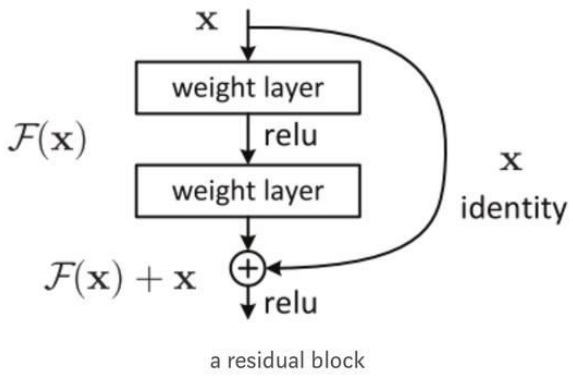


Рисунок 2.5- Залишковий блок ResNet

Це дозволяє складати додаткові шари та будувати більш глибоку мережу, компенсуючи зникаючий градієнт, дозволяючи вашій мережі пропускати через шари, вона вважає, що вони менш актуальні у навчанні.

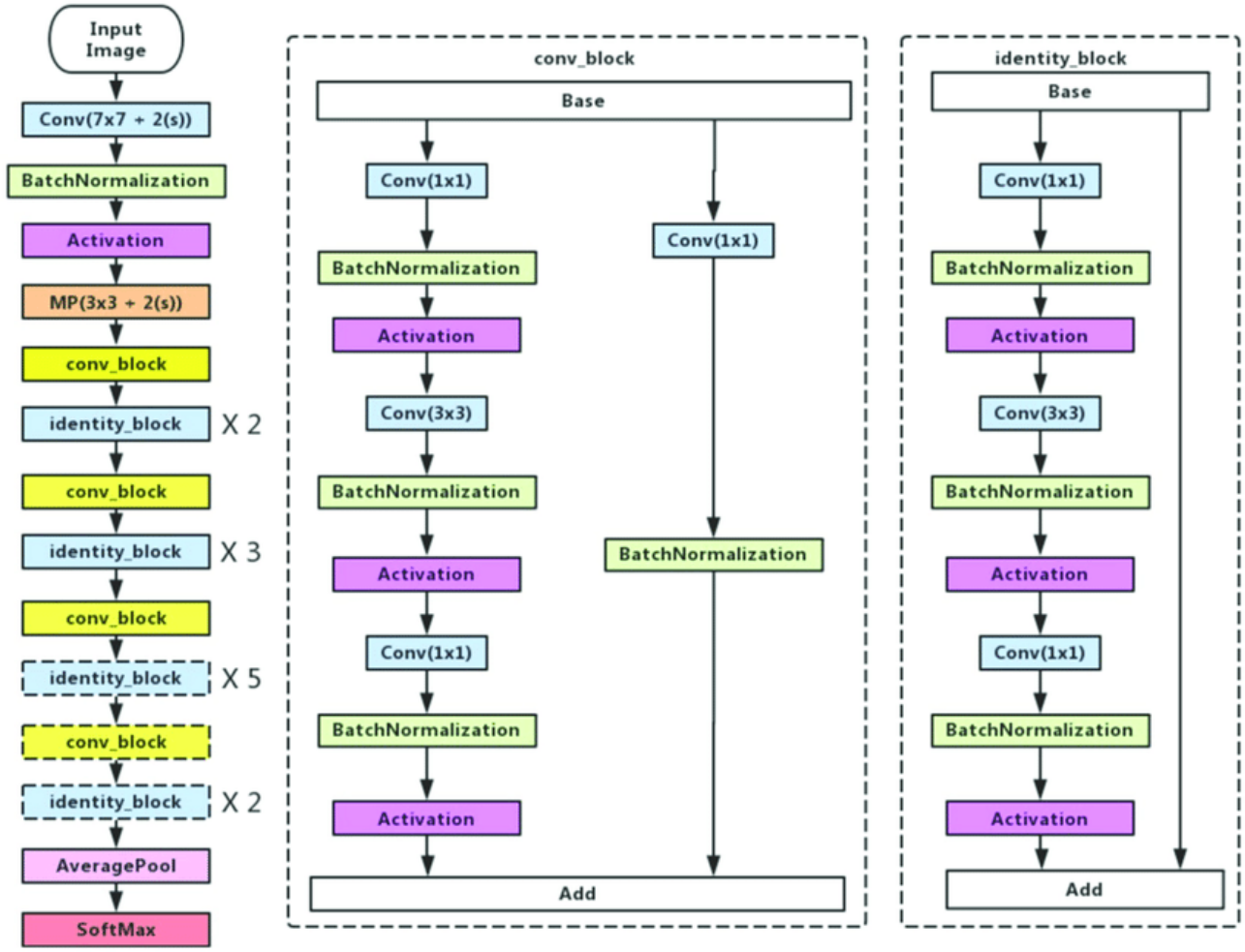


Рисунок 2.6 - Архітектура ResNet-50

Для аугментації я використав горизонтальне повернення з ймовірністю 0.5.

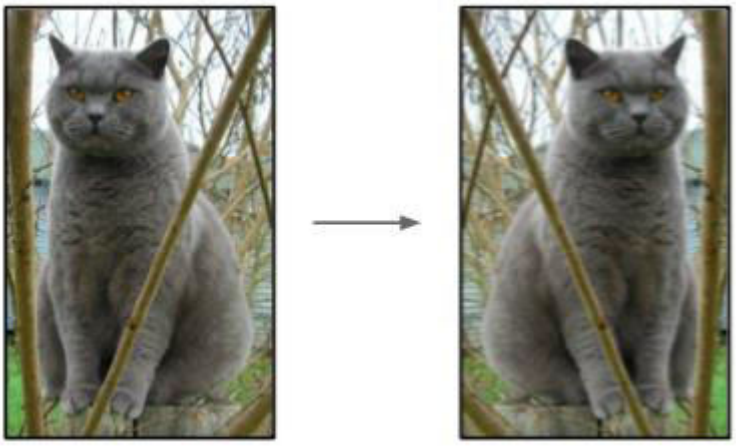


Рисунок 2.7 - Горизонтальне повернення

Порядок навчання я вибрав наступний:

Таблиця 2.2 - Порядок навчання

Етап	Кількість епох	Learning rate	Augmentation
1	2	0.12	Немає
2	4	0.06	Є
3	10	0.012	Є

2.1.6 Експериментальні результати

На Рисунку 2.5 зображений графік validation loss для тренувальної та тестової вибірки.

Найкраще себе показало тренування на 13 епосі, там де validation loss для тестової вибірки найнижчий.

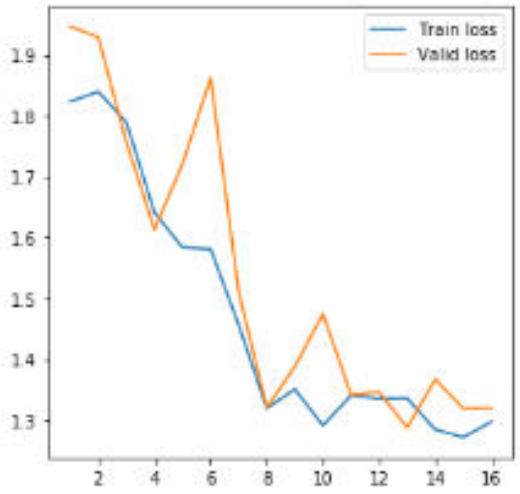


Рисунок 2.8 - Графік Validation Loss для тренувальних та тестових даних

2.1.7 Веб-додаток

Схема, яку я навів містить узагальнені процеси, які проходить користувач під час експлуатації веб додатку. Загальні процеси, які проходить користувач, включають процес завантаження зображення рентген знімка грудної клітки на обробку, обробку зображення та виведення обробленого зображення у форматі PNG.

Процес обробки завантаженого зображення розробленою інформаційною системо зображено за допомогою схеми бізнес-процесу на Рисунку 2.9

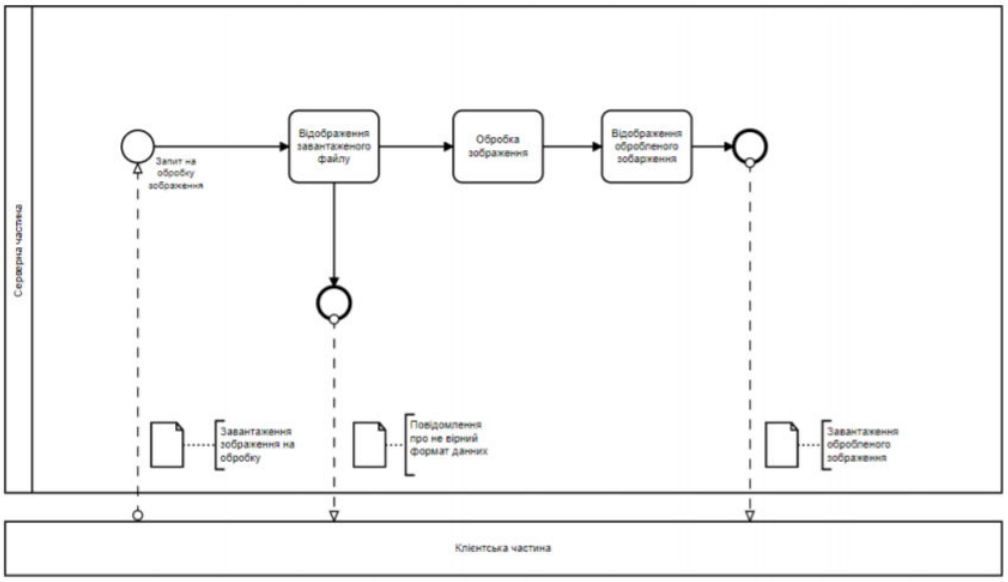


Рисунок 2.9 – Схема бізнес-процесу

Послідовний опис процесу обробки завантаженого зображення наведено нижче.

- Клієнт надсилає ренген знімок грудної клітки у форматі DICOM, яке необхідно обробити.
- Сервер перевіряє завантажений формат завантаженого файлу.
- Якщо завантажений формат не підтримується, то сервер повертає користувача на стартову сторінку.
- Сервер виконує обробку завантаженого зображення.
- Сервер відображає оброблене зображення.

2.2 Архітектура програмного забезпечення

Для вирішення поставленого завдання було обрано клієнт-серверну архітектуру системи. Схема структурна класів програмного забезпечення зображена у документі “Схема структурна класів програмного забезпечення”. Схема структурна розгортання у документі “Схема структурна розгортання”.

Для тренування моделі призначений скрипт train.py. Його методи описано у Таблиці 2.3.

Таблиця 2.3 - Методи train.py

Назва методу	**Аргументи**	**Значення, що повертається**	**Призначення методу**
get_dicom_fps	dicom_dir	list	По розташуванні папки з тестовими зображеннями dicom_dir знаходить всі файли і групує у список
parse_dataset	dicom_dir, anns	image_fps, image_annotations	У папці dicom_dir знаходить всі тестові зображення і групуючи в список image_fps і робить хешмапу з ключем зображення та значеннями – анотації до зображення(список затемнених зон) image_annotations

DetectorConfig – конфігугураційний клас для тренування моделі. Перевизначає значення з суперкласу Config. Його методи описано у таблиці 2.4

Таблиця 2.4 - Методи DetectorConfig

№ п/п	**Назва**	**Значення**	**Пояснення**
1	GPU_COUNT	1	Кількість GPU, які потрібно використовувати

Продовження таблиці 2.4

2	IMAGES_PER_GPU	8	Кількість зображень для тренування на кожному графічному процесорі. Графічний процесор об'ємом 12 Гб зазвичай може обробляти зображення 1024x1024. Налаштування залежно від розміру графічної пам'яті та розмірів зображення.
3	BACKBONE	resnet50	Backbone архітектура мережі Підтримувані значення: resnet50, resnet101.
4	NUM_CLASSES	2	Кількість класів для класифікації(є затемнена зона або немає - 2)
5	RPN_ANCHOR_SCALES	(16,32,64,128)	Довжина сторони квадрата якоря у пікселях
6	TRAIN_ROIS_PER_IMAGE	32	Кількість ROI (Region of Interest) на зображення, яка подається на початок класифікатору, масці.
7	MAX_GT_INSTANCES	4	Максимальна кількість випадків основної істини для використання в одному зображенні
8	DETECTION_MAX_INSTANCES	3	Максимальна кількість остаточних виявлень
9	DETECTION_MIN_CONFIDENCE	0.78	Мінімальне значення ймовірності прийняття виявленого ROI екземпляра нижче цього порогу пропускається
10	DETECTION_NMS_THRESHOLD	0.01	Немаксимальний поріг придушення для виявлення

Продовження таблиці 2.4

11	STEPS_PER_EPOCH	200	Кількість навчальних етапів за епоху. Не обов'язково має відповідати розміру навчального датасету. Оновлення Tenserboard зберігається в кінці кожної епохи, тому зменшення цього значення означає частіші оновлення Tensorboard. Статистика валідації також розраховується на кожній епосі

DetectorDataset – клас, який унаслідує інтерфейс для дата сетів у Matterport Mask R-CNN. Його методи описано у Таблиці 2.5.

Таблиця 2.5 - Методи класу DetectorDataset

Назва методу	**Аргументи**	**Значення, що повертається**	**Призначення методу**
__init__	image_fps, image_annotations, orig_height, orig_weight		Ініціалізація класу
image_reference	image_id	String	Посилання на файл з зображенням у папці
load_image	image_id	np.array	Витягує зображення у форматі np.array по назві зображення

Продовження таблиці 2.5

load_mask	image_id	np.bool, np.array	Витягує по ідентифікатору зображення – чи є в нього затемнена зона і список цих зон у форматі np.array

З набору тренувальних моделей, обирається та, яка дасть найкращу оцінку по Validation Loss. Тоді вона буде використовуватись у веб додатку. Для передбачення результату в сервісу розроблено клас Detector. Його методи описано у Таблиці 2.6.

Таблиця 2.6 - Методи класу Detector

Назва методу	**Аргументи**	**Значення, що повертається**	**Призначення методу**
predict	dicom	File	Запускається процес виявлення затемнених зон. Потім по кожному знайденому екземпляру накладається на вхідне зображення квадрати. Отримане зображення у форматі np.array обробляється в методі convert_to_png і повертається з методу.
convert_to_png	np_array	file	Конвертація np.array у файл з розширення png.

Для взаємодії з кінцевим користувачем розроблено сервіс, написаний на Flask. Його методи вказано у Таблиці 2.7

Таблиця 2.7 - Методи веб сервісу

Назва методу	**Аргументи**	**Значення, що повертається**	**Коди помилок**	**Призначення методу**
upload		response	0x1	Обробка вхідних даних та повернення response

Таблиця 2.8 - Помилки

Код помилки	**Системна назва**	**Значення помилки**
0x1	ERR_INVALID_FORMAT	Файл з вхідними даними має невірний формат

2.3 Конструювання програмного забезпечення

Програмне забезпечення було розроблене з використання технологій та фрейморків, які наведено нижче.

- Python.
- Matterport Mask R-CNN.
- Flask.

Давайте розглянемо їх переваги та недоліки для конструювання даного програмного забезпечення.

2.3.1 Python

Python є лідером у галузі Data Science. Ось деякі важливі фактори, чому я віддав перевагу перед іншими інструментами інформатизації даних [9]:

- легкий у навчанні;
- масштабованість;
- вибір бібліотек обробки даних;

- Python-спільнота;
- графіка та візуалізація.

2.3.2 Matterport Mask R-CNN

Це реалізація Mask R-CNN на Python 3, Keras і TensorFlow. Модель генерує обмежувальні поля та маски сегментації для кожного екземпляра об'єкта на зображенні. Він заснований на мережі функціональних пірамід (FPN) та в якості backbone ResNet101[10].

2.3.3 Flask

В розробці бекенду на Python мені потрібен був фреймворк, з яким я міг швидко ознайомитись і написати мінімальний функціонал.

Flask призначений для швидкого та легкого початку роботи з можливістю масштабування до складних програм. Він розпочався як простий обгортка навколо Werkzeug та Jinja і став однією з найпопулярніших рамок веб-додатків Python. [11]

Flask має властивості, які наведені нижче в списку [12].

- Містить сервер для розробки та відлагоджувач.
- Вбудована підтримка юніт-тестів.
- Управління запитами RESTful.
- Використовує шаблони Jinja2.
- Має підтримку безпечних куків (сесії на стороні клієнта).
- 100% відповідність WSGI 1.0.
- Підтримка Unicode.
- Докладна документація.
- Сумісність з Google App Engine.
- Наявність розширень для забезпечення бажаної поведінки.

2.4 Аналіз безпеки даних

У розроблюваному програмному забезпечення відсутня база даних і отже жодна інформація про користувачів або їхня історія активності не може бути викрадена, тому єдині спроби по викраденню або заміні користувацької інформації можуть мати місце лише під час передачі інформації від сервера до користувача і навпаки. Захист даних користувача гарантує протокол обміну REST, який працює поверх HTTPS та вважається захищеним протоколом обміну інформацією.

2.5 Висновки по розділу

У цьому розділі були описані формати вхідних та вихідних даних. Продемонстровано датасет для навчання нейронної мережі. Стисло описано алгоритм навчання Mask R-CNN. Проаналізовано бізнес-процеси інформаційної системи та проілюстровано з використанням діаграм BPMN. Описано переваги та недоліки обраного набору засобів розробки. Проаналізовано безпеку користувацьких даних в інформаційній системі.

3 ВИПРОБУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Мета випробовувань

Тестування розробленого функціоналу програмного забезпечення дозволить перевірити , чи правильно реалізовано усі вимоги до програмного забезпечення.

Будуть протестовані наступні компоненти розробленого програмного забезпечення:

- тестування алгоритму обробки зображення;
- тестування API сервера по обробці зображень.

3.2 Результати тренування моделі

Було проведено 2 тести. Обидва перевірятимуть точність роботи алгоритму.

Таблиця 3.1 - Тест на середню похибку за формулою Validation Loss

Мета тесту	Перевірка точності роботи алгоритму
Початковий стан	Система готова до обробки зображень. Тестові зображення завантажено в файлову систему
Вхідні дані	Тестові зображення
Схема проведення тесту	Провести тренування моделі, отримати validation loss
Очікуваний результат	Mask R-CNN Validation loss < 1.4
Фактичний результат	Mask R-CNN Validation loss = 1.2878

Validation Loss розраховується за формулою, де $\mathcal{L}$ - функція втрат.

$$Validation\ Loss = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\hat{y}_i, y_i)$$

У алгоритмах, в основі яких лежить згорткова нейронна мережа, використовується функція багаточленна функція втрат та софтмакс на кожен піксель [13].

Багатозадачна функція втрат Mask R-CNN комбінує функції втрат класифікації, локалізації та сегментації масок.

$$\mathcal{L} = \mathcal{L}_{classification} + \mathcal{L}_{mask} + \mathcal{L}_{box}$$

$\mathcal{L}_{mask}$ визначається як середнє значення бінарної кросс-ентропії, включаючи k-ту маску, якщо область пов'язана з основним класом істини k.

$\mathcal{L}_{classification}$ та $\mathcal{L}_{box}$ беруться з алгоритму Faster R-CNN.

Розглянутий тест відновиться до тестів базової функціональності та виконується за допомогою графічного інтерфейсу користувача. Його результати можна перевірити в кінці тренування моделі. Під час проведення тесту слід перевіряти чи співпадає отриманий результат x з очікуваним результатом. Результат виконання тесту наведеного в Таблиці 3.1 зображено на Рисунках 3.1-5.

```
Epoch 1/2
200/200 [==============================] - 685s 3s/step - loss: 1.8235 - rpn_class_loss: 0.0
278 - rpn_bbox_loss: 0.5758 - mrcnn_class_loss: 0.2673 - mrcnn_bbox_loss: 0.5413 - mrcnn_mas
k_loss: 0.4114 - val_loss: 1.9469 - val_rpn_class_loss: 0.0315 - val_rpn_bbox_loss: 0.6938 -
val_mrcnn_class_loss: 0.2143 - val_mrcnn_bbox_loss: 0.5970 - val_mrcnn_mask_loss: 0.4103
Epoch 2/2
200/200 [==============================] - 393s 2s/step - loss: 1.8398 - rpn_class_loss: 0.0
256 - rpn_bbox_loss: 0.7011 - mrcnn_class_loss: 0.2383 - mrcnn_bbox_loss: 0.4827 - mrcnn_mas
k_loss: 0.3921 - val_loss: 1.9292 - val_rpn_class_loss: 0.0238 - val_rpn_bbox_loss: 0.8689 -
val_mrcnn_class_loss: 0.2006 - val_mrcnn_bbox_loss: 0.4476 - val_mrcnn_mask_loss: 0.3882
CPU times: user 6min 22s, sys: 15 s, total: 6min 37s
```

Рисунок 3.1- Епохи 1-2

```
Epoch 3/6
200/200 [==============================] - 1192s 6s/step - loss: 1.7892 - rpn_class_loss: 0.
0225 - rpn_bbox_loss: 0.6300 - mrcnn_class_loss: 0.2624 - mrcnn_bbox_loss: 0.4648 - mrcnn_ma
sk_loss: 0.4095 - val_loss: 1.7566 - val_rpn_class_loss: 0.0241 - val_rpn_bbox_loss: 0.6242
- val_mrcnn_class_loss: 0.2720 - val_mrcnn_bbox_loss: 0.4509 - val_mrcnn_mask_loss: 0.3853
Epoch 4/6
200/200 [==============================] - 898s 4s/step - loss: 1.6409 - rpn_class_loss: 0.0
205 - rpn_bbox_loss: 0.5286 - mrcnn_class_loss: 0.2478 - mrcnn_bbox_loss: 0.4427 - mrcnn_mas
k_loss: 0.4014 - val_loss: 1.6132 - val_rpn_class_loss: 0.0154 - val_rpn_bbox_loss: 0.4998 -
val_mrcnn_class_loss: 0.2535 - val_mrcnn_bbox_loss: 0.4594 - val_mrcnn_mask_loss: 0.3850
Epoch 5/6
200/200 [==============================] - 882s 4s/step - loss: 1.5838 - rpn_class_loss: 0.0
189 - rpn_bbox_loss: 0.4927 - mrcnn_class_loss: 0.2361 - mrcnn_bbox_loss: 0.4372 - mrcnn_mas
k_loss: 0.3988 - val_loss: 1.7213 - val_rpn_class_loss: 0.0212 - val_rpn_bbox_loss: 0.6376 -
val_mrcnn_class_loss: 0.2659 - val_mrcnn_bbox_loss: 0.4260 - val_mrcnn_mask_loss: 0.3706
Epoch 6/6
200/200 [==============================] - 843s 4s/step - loss: 1.5811 - rpn_class_loss: 0.0
177 - rpn_bbox_loss: 0.4905 - mrcnn_class_loss: 0.2478 - mrcnn_bbox_loss: 0.4295 - mrcnn_mas
k_loss: 0.3956 - val_loss: 1.8624 - val_rpn_class_loss: 0.0204 - val_rpn_bbox_loss: 0.7113 -
val_mrcnn_class_loss: 0.2948 - val_mrcnn_bbox_loss: 0.4536 - val_mrcnn_mask_loss: 0.3823
CPU times: user 10min 32s, sys: 28.9 s, total: 11min 1s
```

Рисунок 3.2 - Епохи 3-6

```
Epoch 7/16
200/200 [==============================] - 1304s 7s/step - loss: 1.4564 - rpn_class_loss: 0.
0147 - rpn_bbox_loss: 0.4324 - mrcnn_class_loss: 0.2282 - mrcnn_bbox_loss: 0.3946 - mrcnn_ma
sk_loss: 0.3865 - val_loss: 1.5107 - val_rpn_class_loss: 0.0172 - val_rpn_bbox_loss: 0.5468
- val_mrcnn_class_loss: 0.1834 - val_mrcnn_bbox_loss: 0.3893 - val_mrcnn_mask_loss: 0.3740
Epoch 8/16
200/200 [==============================] - 845s 4s/step - loss: 1.3194 - rpn_class_loss: 0.0
121 - rpn_bbox_loss: 0.3550 - mrcnn_class_loss: 0.2047 - mrcnn_bbox_loss: 0.3688 - mrcnn_mas
k_loss: 0.3788 - val_loss: 1.3205 - val_rpn_class_loss: 0.0115 - val_rpn_bbox_loss: 0.3690 -
val_mrcnn_class_loss: 0.1818 - val_mrcnn_bbox_loss: 0.3869 - val_mrcnn_mask_loss: 0.3712
Epoch 9/16
200/200 [==============================] - 862s 4s/step - loss: 1.3510 - rpn_class_loss: 0.0
130 - rpn_bbox_loss: 0.3763 - mrcnn_class_loss: 0.1981 - mrcnn_bbox_loss: 0.3799 - mrcnn_mas
k_loss: 0.3837 - val_loss: 1.3884 - val_rpn_class_loss: 0.0137 - val_rpn_bbox_loss: 0.4736 -
val_mrcnn_class_loss: 0.1666 - val_mrcnn_bbox_loss: 0.3689 - val_mrcnn_mask_loss: 0.3655
Epoch 10/16
200/200 [==============================] - 834s 4s/step - loss: 1.2912 - rpn_class_loss: 0.0
112 - rpn_bbox_loss: 0.3372 - mrcnn_class_loss: 0.2008 - mrcnn_bbox_loss: 0.3628 - mrcnn_mas
k_loss: 0.3793 - val_loss: 1.4747 - val_rpn_class_loss: 0.0149 - val_rpn_bbox_loss: 0.5275 -
val_mrcnn_class_loss: 0.1830 - val_mrcnn_bbox_loss: 0.3822 - val_mrcnn_mask_loss: 0.3670
Epoch 11/16
200/200 [==============================] - 856s 4s/step - loss: 1.3405 - rpn_class_loss: 0.0
126 - rpn_bbox_loss: 0.3937 - mrcnn_class_loss: 0.1865 - mrcnn_bbox_loss: 0.3695 - mrcnn_mas
k_loss: 0.3781 - val_loss: 1.3417 - val_rpn_class_loss: 0.0116 - val_rpn_bbox_loss: 0.4111 -
val_mrcnn_class_loss: 0.1599 - val_mrcnn_bbox_loss: 0.3916 - val_mrcnn_mask_loss: 0.3675
```

Рисунок 3.3 - Епохи 7-11

```
Epoch 12/16
200/200 [==============================] - 876s 4s/step - loss: 1.3354 - rpn_class_loss: 0.0
126 - rpn_bbox_loss: 0.3887 - mrcnn_class_loss: 0.1802 - mrcnn_bbox_loss: 0.3750 - mrcnn_mas
k_loss: 0.3789 - val_loss: 1.3462 - val_rpn_class_loss: 0.0132 - val_rpn_bbox_loss: 0.3850 -
val_mrcnn_class_loss: 0.1991 - val_mrcnn_bbox_loss: 0.3879 - val_mrcnn_mask_loss: 0.3611
Epoch 13/16
200/200 [==============================] - 893s 4s/step - loss: 1.3359 - rpn_class_loss: 0.0
123 - rpn_bbox_loss: 0.3867 - mrcnn_class_loss: 0.1848 - mrcnn_bbox_loss: 0.3737 - mrcnn_mas
k_loss: 0.3782 - val_loss: 1.2878 - val_rpn_class_loss: 0.0119 - val_rpn_bbox_loss: 0.3983 -
val_mrcnn_class_loss: 0.1559 - val_mrcnn_bbox_loss: 0.3656 - val_mrcnn_mask_loss: 0.3560
Epoch 14/16
200/200 [==============================] - 873s 4s/step - loss: 1.2846 - rpn_class_loss: 0.0
107 - rpn_bbox_loss: 0.3550 - mrcnn_class_loss: 0.1801 - mrcnn_bbox_loss: 0.3656 - mrcnn_mas
k_loss: 0.3731 - val_loss: 1.3671 - val_rpn_class_loss: 0.0133 - val_rpn_bbox_loss: 0.4384 -
val_mrcnn_class_loss: 0.1865 - val_mrcnn_bbox_loss: 0.3684 - val_mrcnn_mask_loss: 0.3605
Epoch 15/16
200/200 [==============================] - 904s 5s/step - loss: 1.2720 - rpn_class_loss: 0.0
106 - rpn_bbox_loss: 0.3527 - mrcnn_class_loss: 0.1779 - mrcnn_bbox_loss: 0.3537 - mrcnn_mas
k_loss: 0.3771 - val_loss: 1.3194 - val_rpn_class_loss: 0.0109 - val_rpn_bbox_loss: 0.3935 -
val_mrcnn_class_loss: 0.1702 - val_mrcnn_bbox_loss: 0.3747 - val_mrcnn_mask_loss: 0.3700
Epoch 16/16
200/200 [==============================] - 805s 4s/step - loss: 1.2974 - rpn_class_loss: 0.0
105 - rpn_bbox_loss: 0.3573 - mrcnn_class_loss: 0.1933 - mrcnn_bbox_loss: 0.3598 - mrcnn_mas
k_loss: 0.3765 - val_loss: 1.3199 - val_rpn_class_loss: 0.0102 - val_rpn_bbox_loss: 0.3984 -
val_mrcnn_class_loss: 0.1894 - val_mrcnn_bbox_loss: 0.3567 - val_mrcnn_mask_loss: 0.3651
CPU times: user 18min 1s, sys: 1min 4s, total: 19min 5s
```

Рисунок 3.4 - Епохи 12-16

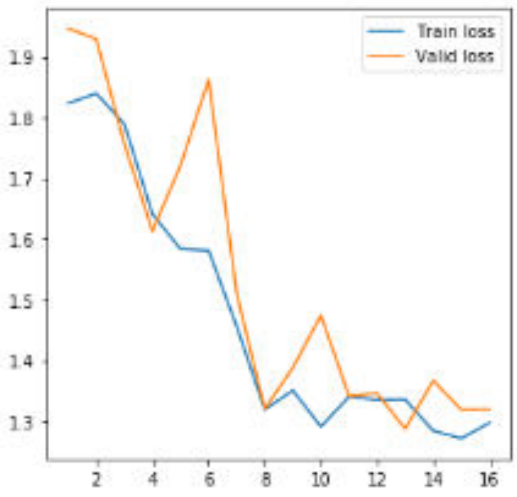


Рисунок 3.5 - Графік validation loss

Як видно, на епосі 13 validation loss приймає найнижче значення 1.2878.

Accuracy оцінюється за середньою середньою точністю при різному порозі перетину над об'єднанням (IoU) [14]. IoU набору прогнозованих обмежувальних коробок і обмежувальних ідентифікаційних полів обчислюється як:

$$IoU(A, B) = \frac{A \cap B}{A \cup B}$$

Метричний показник охоплює діапазон порогів IoU, в кожній точці обчислюючи середнє значення точності. Порогові значення коливаються від 0,4 до 0,75 із розміром кроку 0,05: (0,4, 0,45, 0,5, 0,55, 0,6, 0,65, 0,7, 0,75). Іншими словами, при порозі 0,5 прогнозований об'єкт вважається "успішним", якщо його перетин над об'єднанням із основним об'єктом істини перевищує 0,5.

При кожному пороговому значенні t значення точності обчислюється виходячи з кількості вірних позитивних (TP), хибних негативних (FN) та хибних позитивних результатів (FP), отриманих в результаті порівняння передбачуваного об'єкта з усіма об'єктами первинної істини:

$$\frac{TP(t)}{TP(t) + FP(t) + FN(t)}$$

Середня точність одного зображення обчислюється як середнє значення вищевказаних значень точності на кожному порозі IoU:

$$\frac{1}{N}\sum_{t}\frac{TP(t)}{TP(t) + FP(t) + FN(t)}$$

IoU, Intersection Over Union - це міра величини перекриття між двома обмежувальними полями (або, у більш загальному випадку, двома об’єктами). Він обчислює розмір перекриття між двома об’єктами, поділений на загальну площу двох об'єднаних об’єднаних. Наглядно зображено на Рисунку 3.6.

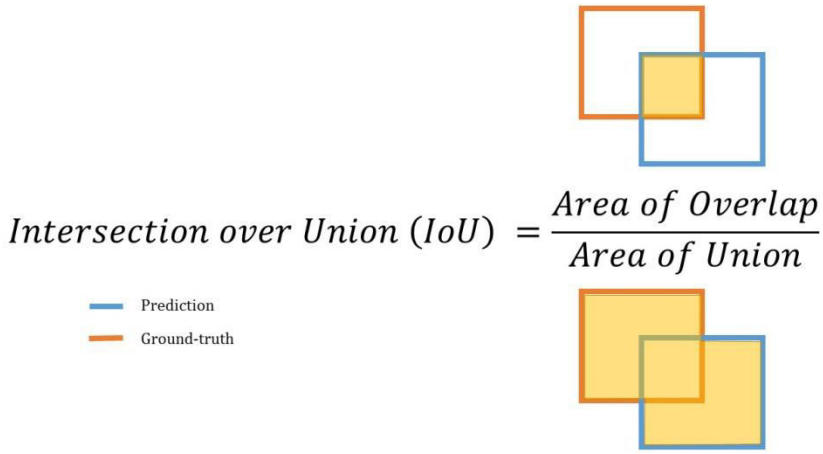


Рисунок 3.6 – Intersection Over Union

Результат виконання тесту наведеного в Таблиці 3.1 зображено в Таблиці 3.2 та на Рисунку 3.7.

Таблиця 3.2 - Тест на середню похибку за формулою Accuracy

Мета тесту	Перевірка точності роботи алгоритму
Початковий стан	Отримано натреновану модель
Вхідні дані	Ваги натренованої моделі
Схема проведення тесту	Запустити скрипт знаходження результату для 3000 тестових зображень у форматі csv. Завантажити отриманий файл у систему для перевірки точності Kaggle.
Очікуваний результат	Accuracy не менше 0.1
Фактичний результат	Accuracy = 0.1

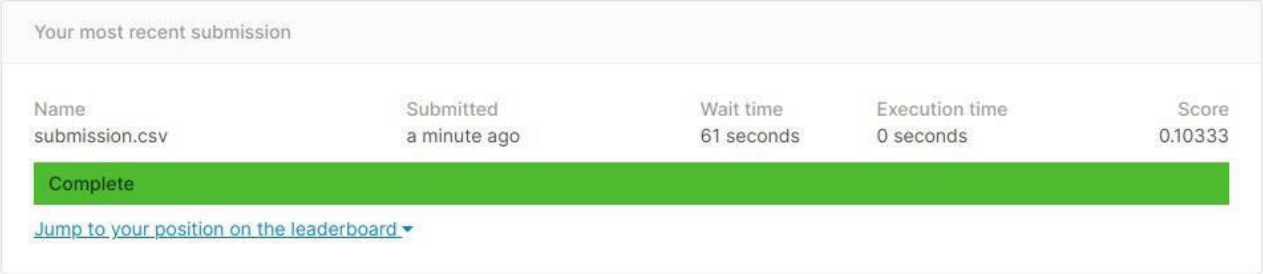


Рисунок 3.7 - Результат виконання тесту на середню похибку за формулою Accuracy

3.3 Результати роботи веб сервісу

Функції, що необхідно протестувати в роботі веб сервісу:

- негативний тест на обробку файлу з неправильним форматом;
- тест REST сервіса;
- час виконання запиту на обробку.

Результати тестування зображено в Таблицях 3.3-3.5 та Рисунку 3.8.

Таблиця 3.3 - Негативний тест на обробку файлу з неправильним форматом

Мета тесту	Перевірка роботи веб сервісу
Початковий стан	Система готова до обробки файлу. Файл “poltava.pdf” завантажено в систему
Вхідні дані	Файл “poltava.pdf”
Схема проведення тесту	Запустити скрипт обробки файлу та обрати файл “poltava.pdf” як вхідне зображення
Очікуваний результат	Після проведення тесту стан системи повернувся до початкового
Фактичний результат	Після проведення тесту стан системи повернувся до початкового

Таблиця 3.4 - Тест на роботу REST сервісу

Мета тесту	Перевірка роботи веб сервісу
Початковий стан	Система готова до обробки файлу. Система для тестування Postman підготовлена. Налаштовано url, cookies та обрано вхідний файл “001b0c51-c7b3-45c1-9c17-fa7594cab96e.dcm”
Вхідні дані	Файл “001b0c51-c7b3-45c1-9c17-fa7594cab96e.dcm”
Схема проведення тесту	Запустити скрипт обробки файлу та обрати файл “001b0c51-c7b3-45c1-9c17-fa7594cab96e.dcm” як вхідне зображення
Очікуваний результат	Відповіддю REST сервіса є зображення з затемненими зонами
Фактичний результат	Відповіддю REST сервіса є зображення з затемненими зонами

Таблиця 3.5- Тестування часу обробки зображення

Мета тесту	Перевірка роботи веб сервісу, часу обробки зображення
Початковий стан	Система готова до обробки файлу. Система для тестування Postman підготовлена. Налаштовано url, cookies та обрано вхідний файл “001b0c51-c7b3-45c1-9c17-fa7594cab96e.dcm”
Вхідні дані	Файл “001b0c51-c7b3-45c1-9c17-fa7594cab96e.dcm”
Схема проведення тесту	Запустити скрипт обробки файлу та обрати файл “001b0c51-c7b3-45c1-9c17-fa7594cab96e.dcm” як вхідне зображення
Очікуваний результат	Час виконання запиту менше 3 с
Фактичний результат	Час виконання запиту 1361 мс

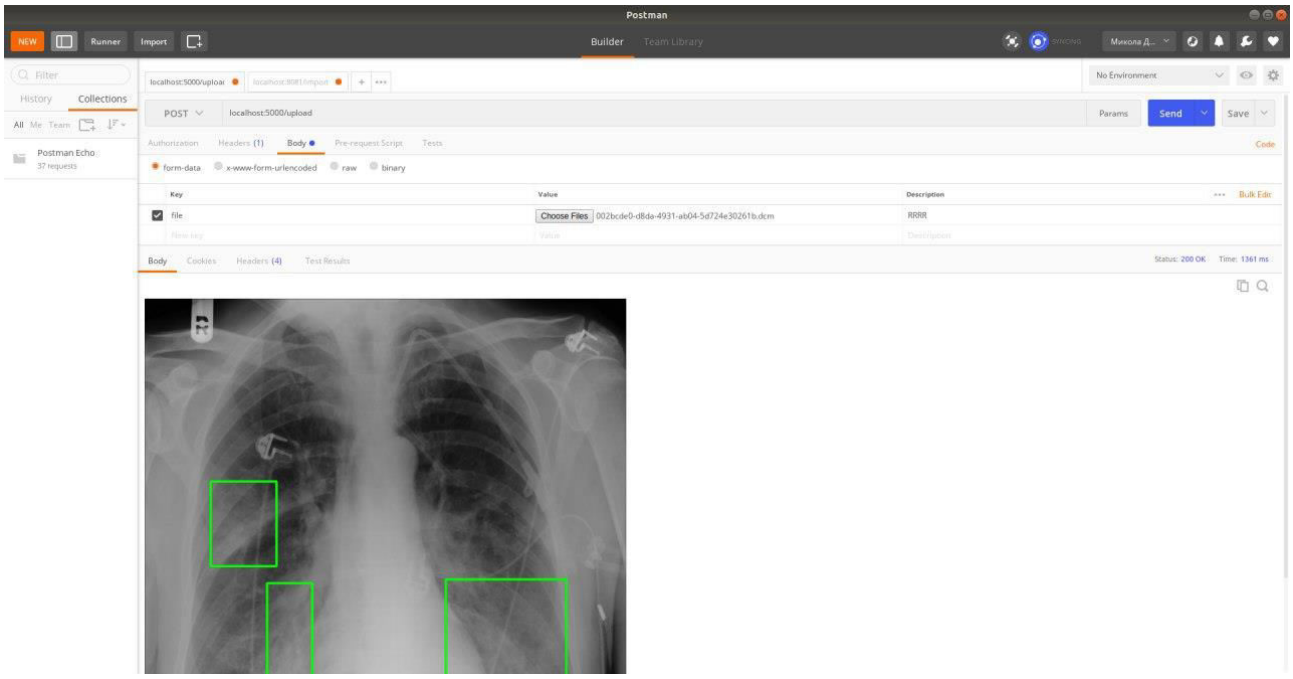


Рисунок 3.8 - Тест роботи веб сервісу

3.4 Висновок по розділу

По обох метриках результат тренування моделі задовільний. Покращити його можна за допомогою використання потужнішою обчислювальної техніки та провівши роботу на оптимізацію гіперпараметрів. У своїй роботі я їх підбирав згідно рекомендацій з інших робіт.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання розробленого програмного забезпечення необхідно виконати наступні кроки.

1) Інсталювати Python 3.7 або вище в системі.

2) Запустити сервер по обробці даних виконавши команду «python3 app.py».

4.2 Робота з програмним забезпеченням

Розроблене програмне забезпечення передбачає наступний сценарій використання.

1) Відкриття застосунку в браузері.

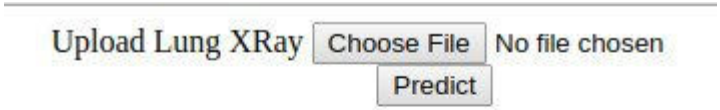


Рисунок 4.1 – Стартова сторінка

2) Натиснути на кнопку “Choose File”.

3) Обрати необхідний файл в файловому провіднику (Рисунок 4.2).

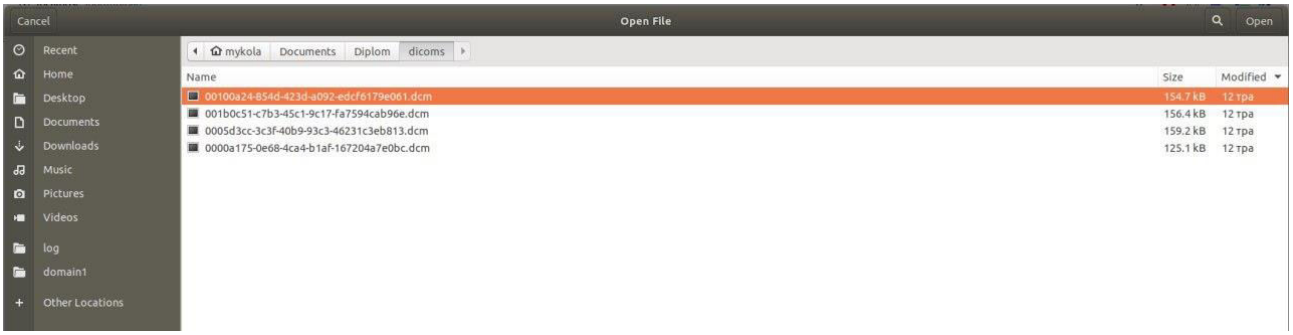


Рисунок 4.2 - Вибір зображення в файловому провіднику

4) Натиснути кнопку “Predict”.

Після цього зображення буде передано на сервер для його обробки. Через декілька секунд користувачеві буде відображено результат роботи програмного забезпечення.

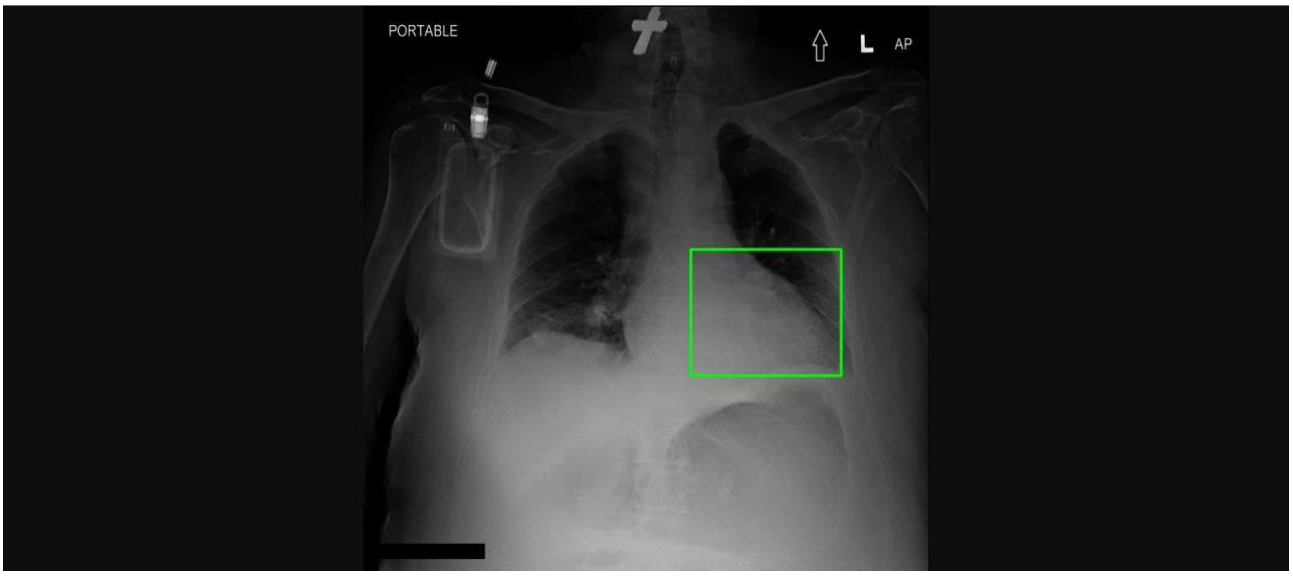


Рисунок 4.3 - Результат роботи

4.3 Висновки по розділу

У даному розділі продемонстровано схему розгортання системи та описано сценарій використання розробленої інформаційної системи.

ВИСНОВКИ

У ході дипломного проекту був проведений аналіз передбачення затемненених зон на рентген знімку грудної клітки, описано відомі технічні рішення. На основі досліджень розроблений програмний комплекс для вирішення цієї проблеми.

В цій роботі було розроблене програмне забезпечення для тренування моделей машинного навчання і програмне забезпечення для вирішення задачі. Для демонстрації навчених моделей був створений веб-застосунок.

Розроблено необхідну проектну документацію, схеми процесів застосунку, варіантів використань та керівництво користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1) Машинне навчання: Вікіпедія. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%B5_%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%8F

2) Felson's Principles of Chest Roentgenology (Fourth Edition).

3) Pneumonia: Jama - 2016. – Режим доступу до ресурсу: https://jamanetwork.com/journals/jama/fullarticle/2488310.

4) CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning – 2017. – Режим доступу до ресурсу: https://arxiv.org/abs/1711.05225

5) Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases – 2017. – Режим доступу до ресурсу: https://arxiv.org/abs/1705.02315

6) RetinaNet Pneumonia Detection: Dmytro Poplavskiy – 2018. – Режим доступу до ресурсу: https://github.com/pdima/kaggle_RSNA_Pneumonia_Detection

7) Pneumonia Detection: Ian Pan – 2018. – Режим доступу до ресурсу: https://github.com/i-pan/kaggle-rsna18

8) ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. – Режим доступу до ресурсу: http://openaccess.thecvf.com/content_cvpr_2017/papers/Wang_ChestX-ray8_Hospital-Scale_Chest_CVPR_2017_paper.pdf

9) Python Official Site. – Режим доступу до ресурсу: https://www.python.org/

10) Matterport Mask R-CNN. - Режим доступу до ресурсу: https://github.com/matterport/Mask_RCNN

11) Flask Documentation – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/1.1.x/

12) Flask: Вікіпедія. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Flask

13) What is the loss function of the Mask RCNN? : Stack Overflow – 2017. – Режим доступу до ресурсу: https://stackoverflow.com/questions/46272841/what-is-the-loss-function-of-the-mask-rcnn

14) Accuracy Evaluation. – Режим доступу до ресурсу: https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/overview/evaluation

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр Павлов

“___” _____ 2020 р.

«Програмне застосування для виявлення затемнених зон на рентген знімку легень»

Технічне завдання

КПІ.ІП-6107.045490.03.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О. П. Сирота

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ М. І. Данилюк

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: «Програмне застосування для виявлення затемнених зон на рентген знімку легень».

Галузь застосування: програмне застосування для діагностики пневмонії.

Наведене технічне завдання поширюється на розробку «Програмне застосування для виявлення затемнених зон на рентген знімку легень» КПІ.ІП-6107.045490.03.91, котра використовується для систем діагностики пневмонії.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки «Програмне застосування для виявлення затемнених зон на рентген знімку легень» є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в системах діагностики пневмонії.

Метою розробки є створення програмного застосування для виявлення затемнених зон на рентген знімку легень.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання перетворення зображення DICOM рентген знімка грудної клітки у зображення PNG з наведеними зонами затемнення. Розробку виконати на платформі Linux.

4.2 Вимоги до надійності

Програмне забезпечення повинно передбачити контроль введення інформації та захист від некоректних дій користувача.

4.3 Умови експлуатації

Програмне забезпечення повинно відповідати умовам експлуатації згідно СанПін 2.2.2.542 – 96. Розроблюване програмне забезпечення функціонує в автоматичному режимі та не вимагає обслуговування.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах. Мінімальна конфігурація технічних засобів:

- центральний процесор: 1.4 GHz Quad-Core 8-th generation Intel Core i5;
- оперативна пам’ять: 8 GB;
- постійна пам’ять: 256 GB.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Unix. Вхідні дані повинні бути представлені в наступному форматі: зображення грудної клітки у форматі DICOM. Результати повинні бути представлені в наступному форматі: ображення PNG з

наведеними зонами затемнення. Програмне забезпечення повинно бути розроблено на мові програмування Python з використанням фреймворків Mask R-CNN та Flask в IDE PyCharm.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему.

5.3 У склад супроводжувальної документації повинні входити наступні документи:

- пояснювальна записка не менше ніж на 50 аркушах формату А4 (без додатків 5.3.2 - 5.3.6);
- технічне завдання;
- керівництво користувача;
- графічна частина повинна бути виконана на 3 листах формату А3, котрі включаються у якості додатків до пояснювальної записки: «Структура генеративної моделі», «Схема класів програмного забезпечення», «Схема структурна розгортання».

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

	Назва етапу	Строк	Звітність
1	Вивчення літератури за тематикою проекту	10.04.20	
2	Розробка технічного завдання	15.04.20	Технічне завдання
3	Аналіз вимог та уточнення специфікацій	17.04.20	Специфікації програмного забезпечення
4	Проектування структури програмного забезпечення, проектування компонентів	20.04.20	Схема структурна програмного забезпечення та специфікація компонентів
5	Програмна реалізація програмного забезпечення	01.05.20	Вихідні тексти програмного забезпечення
6	Тестування програмного забезпечення	08.05.20	Тести, результати тестування
7	Розробка матеріалів текстової частини проекту	15.05.20	Пояснювальна записка.
8	Розробка матеріалів графічної частини проекту	15.05.20	Графічний матеріал проекту
9	Оформлення технічної документації проекту	15.05.20	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Будуть протестовані наступні компоненти розробленого програмного забезпечення:

- тестування алгоритму обробки зображення;
- тестування API сервера по обробці зображень.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ПРОГРАМНЕ ЗАСТОСУВАННЯ ДЛЯ ВИЯВЛЕННЯ ЗАТЕМНЕНИХ ЗОН НА РЕНТГЕН ЗНІМКУ ЛЕГЕНЬ

Опис програми

КПІ.ІП-6107.045490.04.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О. П. Сирота

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____М. І. Данилюк

Київ – 2020 року

Тексти програмного коду

Програмне застосування для виявлення затемнених зон на рентген знімку легень

(Наименування програми (документа))

DVD-R

(Вид носія даних)

7 арк, 179 254 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ - 2020

train.py

```
import os
import sys
import random
import math
import numpy as np
import cv2
import matplotlib.pyplot as plt
import json
import pydicom
from imgaug import augmenters as iaa
from tqdm import tqdm
import pandas as pd
import glob
from sklearn.model_selection import KFold

DATA_DIR = '/kaggle/input'

# Directory to save logs and trained model
ROOT_DIR = '/kaggle/working'

!git clone https://www.github.com/matterport/Mask_RCNN.git
os.chdir('Mask_RCNN')
#!python setup.py -q install

# Import Mask RCNN
sys.path.append(os.path.join(ROOT_DIR, 'Mask_RCNN'))  # To find local version of the library
from mrcnn.config import Config
from mrcnn import utils
import mrcnn.model as modellib
from mrcnn import visualize
from mrcnn.model import log

train_dicom_dir = os.path.join(DATA_DIR, 'stage_2_train_images')
test_dicom_dir = os.path.join(DATA_DIR, 'stage_2_test_images')

!wget --quiet https://github.com/matterport/Mask_RCNN/releases/download/v2.0/mask_rcnn_coco.h5
!ls -lh mask_rcnn_coco.h5

COCO_WEIGHTS_PATH = "mask_rcnn_coco.h5"

def get_dicom_fps(dicom_dir):
    dicom_fps = glob.glob(dicom_dir+'/'+'*.dcm')
    return list(set(dicom_fps))

def parse_dataset(dicom_dir, anns):
    image_fps = get_dicom_fps(dicom_dir)
    image_annotations = {fp: [] for fp in image_fps}
    for index, row in anns.iterrows():
        fp = os.path.join(dicom_dir, row['patientId']+'.dcm')
        image_annotations[fp].append(row)
    return image_fps, image_annotations


image_annotations


# The following parameters have been selected to reduce running time for demonstration purposes
# These are not optimal

class DetectorConfig(Config):
    """Configuration for training pneumonia detection on the RSNA pneumonia dataset.
    Overrides values in the base Config class.
```

```
    """

    # Give the configuration a recognizable name
    NAME = 'pneumonia'

    # Train on 1 GPU and 8 images per GPU. We can put multiple images on each
    # GPU because the images are small. Batch size is 8 (GPUs * images/GPU).
    GPU_COUNT = 1
    IMAGES_PER_GPU = 8

    BACKBONE = 'resnet50'

    NUM_CLASSES = 2  # background + 1 pneumonia classes

    IMAGE_MIN_DIM = 256
    IMAGE_MAX_DIM = 256
    RPN_ANCHOR_SCALES = (16, 32, 64, 128)
    TRAIN_ROIS_PER_IMAGE = 32
    MAX_GT_INSTANCES = 4
    DETECTION_MAX_INSTANCES = 3
    DETECTION_MIN_CONFIDENCE = 0.78  ## match target distribution
    DETECTION_NMS_THRESHOLD = 0.01

    STEPS_PER_EPOCH = 200


config = DetectorConfig()
config.display()


class DetectorDataset(utils.Dataset):
    """Dataset class for training pneumonia detection on the RSNA pneumonia dataset.
    """

    def __init__(self, image_fps, image_annotations, orig_height, orig_width):
        super().__init__(self)

        # Add classes
        self.add_class('pneumonia', 1, 'Lung Opacity')

        # add images
        for i, fp in enumerate(image_fps):
            annotations = image_annotations[fp]
            self.add_image('pneumonia', image_id=i, path=fp,
                           annotations=annotations, orig_height=orig_height, orig_width=orig_width)

    def image_reference(self, image_id):
        info = self.image_info[image_id]
        return info['path']

    def load_image(self, image_id):
        info = self.image_info[image_id]
        fp = info['path']
        ds = pydicom.read_file(fp)
        image = ds.pixel_array
        # If grayscale. Convert to RGB for consistency.
        if len(image.shape) != 3 or image.shape[2] != 3:
            image = np.stack((image,) * 3, -1)
        return image

    def load_mask(self, image_id):
        info = self.image_info[image_id]
        annotations = info['annotations']
        count = len(annotations)
```

```
        if count == 0:
            mask = np.zeros((info['orig_height'], info['orig_width'], 1), dtype=np.uint8)
            class_ids = np.zeros((1,), dtype=np.int32)
        else:
            mask = np.zeros((info['orig_height'], info['orig_width'], count), dtype=np.uint8)
            class_ids = np.zeros((count,), dtype=np.int32)
            for i, a in enumerate(annotations):
                if a['Target'] == 1:
                    x = int(a['x'])
                    y = int(a['y'])
                    w = int(a['width'])
                    h = int(a['height'])
                    mask_instance = mask[:, :, i].copy()
                    cv2.rectangle(mask_instance, (x, y), (x + w, y + h), 255, -1)
                    mask[:, :, i] = mask_instance
                    class_ids[i] = 1
        return mask.astype(np.bool), class_ids.astype(np.int32)

# Parse the dataset
image_fps, image_annotations = parse_dataset(train_dicom_dir, anns=anns)

# Original DICOM image size: 1024 x 1024
ORIG_SIZE = 1024

# Split the data into training and validation datasets
image_fps_list = list(image_fps)
random.seed(42)
random.shuffle(image_fps_list)
val_size = 1500
image_fps_val = image_fps_list[:val_size]
image_fps_train = image_fps_list[val_size:]

# prepare the training dataset
dataset_train = DetectorDataset(image_fps_train, image_annotations, ORIG_SIZE, ORIG_SIZE)
dataset_train.prepare()

# prepare the validation dataset
dataset_val = DetectorDataset(image_fps_val, image_annotations, ORIG_SIZE, ORIG_SIZE)
dataset_val.prepare()

# Image augmentation
augmentation = iaa.Sequential([
    iaa.Fliplr(0.5)])

# Now it's time to train the model.
model = modellib.MaskRCNN(mode='training', config=config, model_dir=ROOT_DIR)

# Exclude the last layers because they require a matching number of classes
model.load_weights(COCO_WEIGHTS_PATH, by_name=True, exclude=[
    "mrcnn_class_logits", "mrcnn_bbox_fc",
    "mrcnn_bbox", "mrcnn_mask"])

LEARNING_RATE = 0.006

# Train Mask-RCNN Model
import warnings
warnings.filterwarnings("ignore")

# Step 1
%%time
## train heads with higher lr to speedup the learning
model.train(dataset_train, dataset_val,
            learning_rate=LEARNING_RATE*2,
            epochs=2,
```

```
        layers='heads',
        augmentation=None)  ## no need to augment yet

history = model.keras_model.history.history

# Step 2
%%time
model.train(dataset_train, dataset_val,
        learning_rate=LEARNING_RATE,
        epochs=6,
        layers='all',
        augmentation=augmentation)

new_history = model.keras_model.history.history
for k in new_history: history[k] = history[k] + new_history[k]

# Step 3
%%time
model.train(dataset_train, dataset_val,
        learning_rate=LEARNING_RATE/5,
        epochs=16,
        layers='all',
        augmentation=augmentation)

new_history = model.keras_model.history.history
for k in new_history: history[k] = history[k] + new_history[k]

# DataFrame with history
epochs = range(1,len(next(iter(history.values())))+1)
pd.DataFrame(history, index=epochs)

# Plot with history
plt.figure(figsize=(17,5))

plt.subplot(131)
plt.plot(epochs, history["loss"], label="Train loss")
plt.plot(epochs, history["val_loss"], label="Valid loss")
plt.legend()
plt.subplot(132)
plt.plot(epochs, history["mrcnn_class_loss"], label="Train class ce")
plt.plot(epochs, history["val_mrcnn_class_loss"], label="Valid class ce")
plt.legend()
plt.subplot(133)
plt.plot(epochs, history["mrcnn_bbox_loss"], label="Train box loss")
plt.plot(epochs, history["val_mrcnn_bbox_loss"], label="Valid box loss")
plt.legend()

plt.show()

# Find best epoch
best_epoch = np.argmin(history["val_loss"])
print("Best Epoch:", best_epoch + 1, history["val_loss"][best_epoch])
```

app.py

```
from flask import Flask
from views import bp

app = Flask(__name__)
app.register_blueprint(bp)

if __name__ == '__main__':
    app.secret_key = 'super secret key'
    app.config['SESSION_TYPE'] = 'filesystem'
```

```
    app.debug = True
    app.run()
```

views.py

```
from io import BytesIO

import cv2
import matplotlib.pyplot as plt
import tensorflow as tf
import mrcnn.model as modellib
import numpy as np
import pydicom
from flask import Blueprint
from flask import flash, request, redirect, make_response
from flask import render_template
from mrcnn import utils
from mrcnn.config import Config
from tensorflow.python.keras.backend import set_session

bp = Blueprint('bp', __name__, template_folder='resources/templates')


class DetectorConfig(Config):
    """Configuration for training pneumonia detection on the RSNA pneumonia dataset.
    Overrides values in the base Config class.
    """
    # Give the configuration a recognizable name
    NAME = 'pneumonia'
    # Train on 1 GPU and 8 images per GPU. We can put multiple images on each
    # GPU because the images are small. Batch size is 8 (GPUs * images/GPU).
    GPU_COUNT = 1
    IMAGES_PER_GPU = 8
    BACKBONE = 'resnet50'
    NUM_CLASSES = 2  # background + 1 pneumonia classes
    IMAGE_MIN_DIM = 256
    IMAGE_MAX_DIM = 256
    RPN_ANCHOR_SCALES = (4, 8, 16, 32, 64)
    TRAIN_ROIS_PER_IMAGE = 32
    MAX_GT_INSTANCES = 4
    DETECTION_MAX_INSTANCES = 3
    DETECTION_MIN_CONFIDENCE = 0.78  ## match target distribution
    DETECTION_NMS_THRESHOLD = 0.01
    STEPS_PER_EPOCH = 200


class InferenceConfig(DetectorConfig):
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1


def load_model():
    set_session(sess)
    inference_config = InferenceConfig()
    # Recreate the model in inference mode
    model = modellib.MaskRCNN(mode='inference',
                              config=inference_config,
                              model_dir='resources/working')
    # Load trained weights (fill in path to trained weights here)
    model_path = 'resources/weight.h5'
    print("Loading weights from ", model_path)
    model.load_weights(model_path, by_name=True)
    model.keras_model._make_predict_function()
    return model


tf_config = tf.ConfigProto(
```

```
        device_count={'CPU': 1},
        intra_op_parallelism_threads=1,
        allow_soft_placement=True
    )
    sess = tf.Session(config=tf_config)
    graph = tf.get_default_graph()
    model = load_model()


    def allowed_file(filename):
        return '.' in filename and filename.rsplit('.', 1)[1].lower() == 'dcm'


    @bp.route('/upload', methods=['GET', 'POST'])
    def upload():
        if request.method == 'POST':
            # check if the post request has the file part
            if 'file' not in request.files:
                flash('No file part')
                return redirect(request.url)
            file = request.files['file']
            # if user does not select file, browser also
            # submit an empty part without filename
            if file.filename == '':
                flash('No selected file')
                return redirect(request.url)
            if not allowed_file(file.filename):
                return redirect(request.url)
            image_np = predict(file)
            buf = BytesIO()
            plt.imsave(buf, image_np, format='png')
            response = make_response(buf.getvalue())
            response.headers.set('Content-Type', 'image/png')
            return response
        return render_template('upload.html')


    # IMPORTANT: models have to be loaded AFTER SETTING THE SESSION for keras!
    # Otherwise, their weights will be unavailable in the threads after the session there has
been set
    set_session(sess)


    def predict(dicom):
        config = DetectorConfig()
        ds = pydicom.read_file(dicom)
        # original image
        image = ds.pixel_array
        # assume square image
        resize_factor = 4
        if len(image.shape) != 3 or image.shape[2] != 3:
            image = np.stack((image,) * 3, -1)
        resized_image, window, scale, padding, crop = utils.resize_image(
            image,
            min_dim=config.IMAGE_MIN_DIM,
            min_scale=config.IMAGE_MIN_SCALE,
            max_dim=config.IMAGE_MAX_DIM,
            mode=config.IMAGE_RESIZE_MODE)
        global sess
        global graph
        with graph.as_default():
            set_session(sess)
            results = model.detect([resized_image])
        r = results[0]
        for bbox in r['rois']:
            x1 = int(bbox[1] * resize_factor)
            y1 = int(bbox[0] * resize_factor)
```

```
            x2 = int(bbox[3] * resize_factor)
            y2 = int(bbox[2] * resize_factor)
            cv2.rectangle(image, (x1, y1), (x2, y2), (77, 255, 9), 3, 1)
        return image
```

upload.html

```
    <div class="row" style="text-align: center">
        <div class="col-md-4 col-md-offset-4">
            <form method=POST class="form-inline" enctype=multipart/form-data action="{{
url_for('bp.upload') }}">
                <div class="form-group">
                    <label for="inputFile">Upload Lung XRay</label>
                    <input id="inputFile" type=file  name=file><br />
                    <input class="btn btn-default" type="submit" value="Predict">
                </div>
            </form>
        </div>
    </div>
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр Павлов

“___” _____ 2020 р.

ПРОГРАМНЕ ЗАСТОСУВАННЯ ДЛЯ ВИЯВЛЕННЯ ЗАТЕМНЕНИХ ЗОН НА РЕНТГЕН ЗНІМКУ ЛЕГЕНЬ

Керівництво користувача

КПІ.ІП-6107.045490.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О. П. Сирота

Виконавець:

_____ М. І. Данилюк

Нормоконтроль:

_____ К.І. Ліщук

Київ – 2020 року

Перед початком роботи з програмним продуктом користувач повинен завантажити стартову сторінку веб-додаток (Рисунок 1.1).

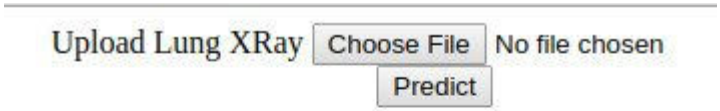


Рисунок 1.1 – Стартова сторінка

Після завантаження стартової сторінки користувачеві необхідно нажати на кнопку “Choose File” та обрати необхідний файл в файловому провіднику (Рисунок 1.2).

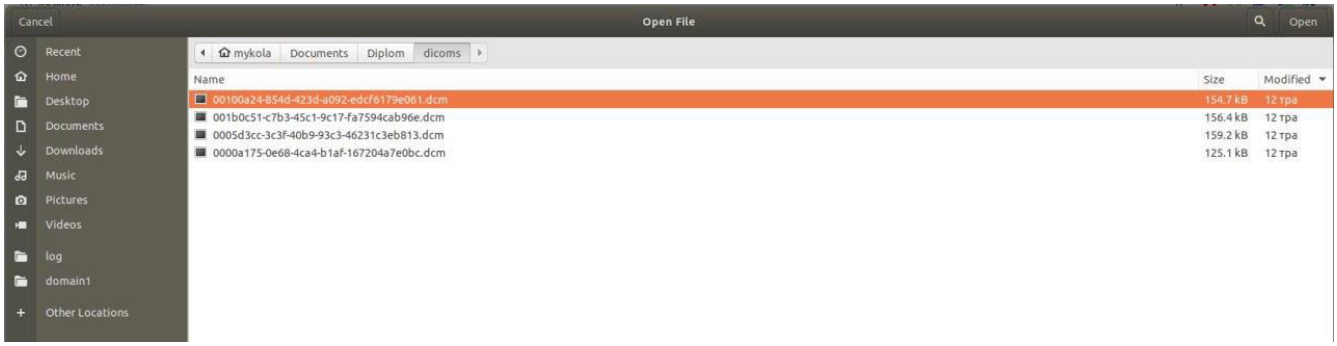


Рисунок 1.2 – Вибір файлу

Після обрання зображення необхідно нажати на кнопку “Predict” (Рисунок 1.3)

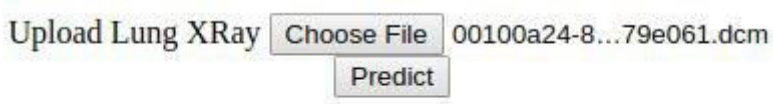


Рисунок 1.3 – Сторінка після обрання зображення

Після цього зображення буде передано на сервер для його обробки. Через декілька секунд користувачеві буде відображено результат роботи програмного забезпечення.

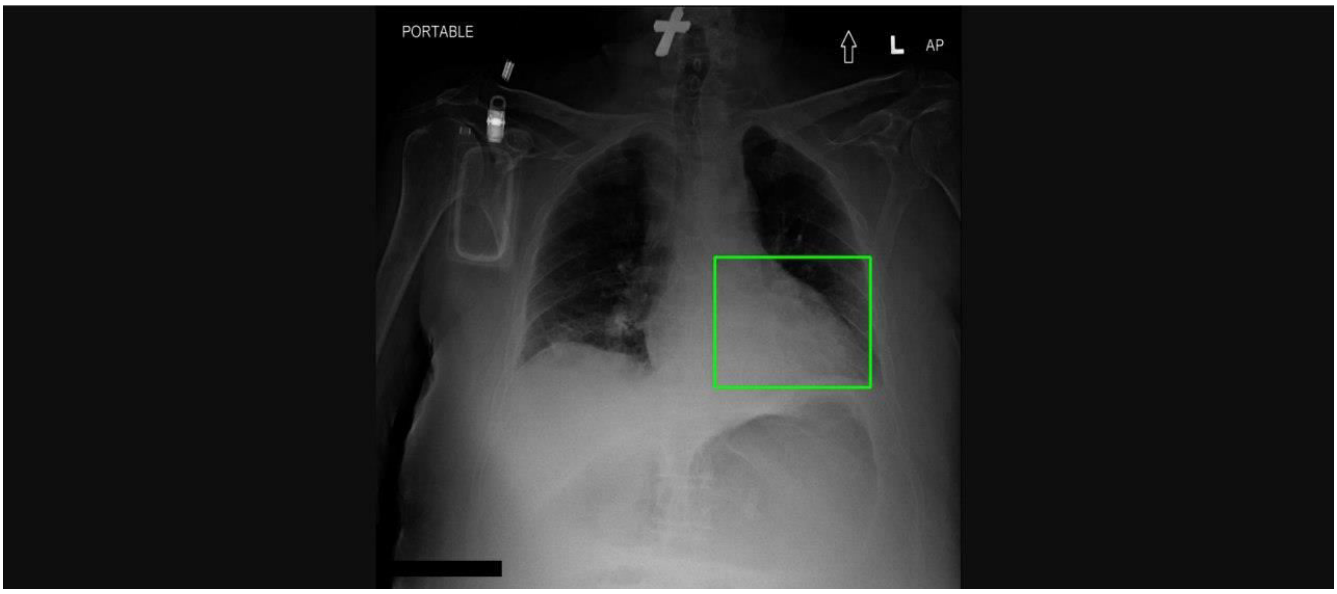


Рисунок 1.4 – Результат

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ПРОГРАМНЕ ЗАСТОСУВАННЯ ДЛЯ ВИЯВЛЕННЯ ЗАТЕМНЕНИХ ЗОН НА РЕНТГЕН ЗНІМКУ ЛЕГЕНЬ

Графічний матеріал

КПІ.ІП-6107.045490.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О. П. Сирота

Виконавець:

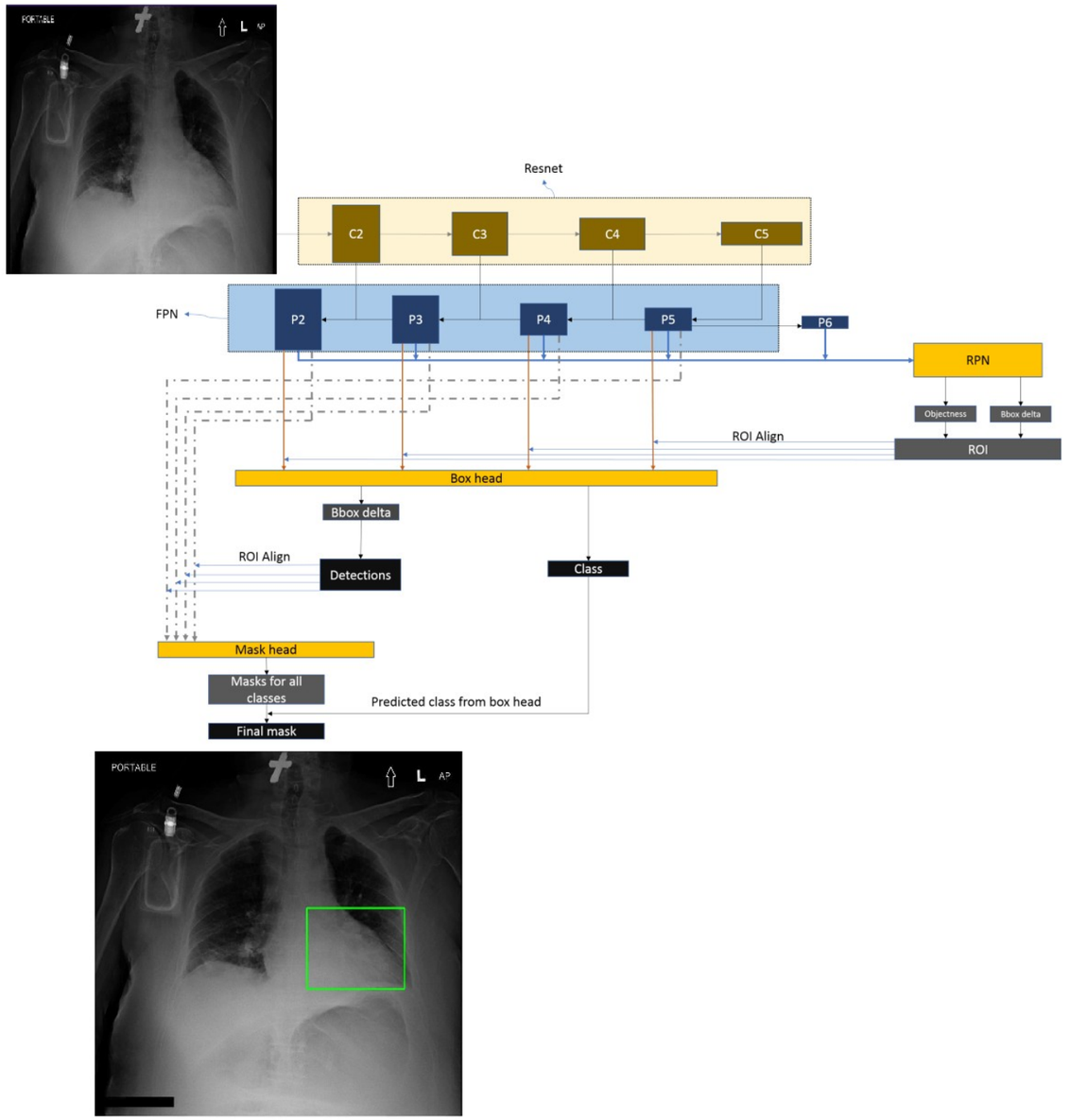
_____М. І. Данилюк

Нормоконтроль:

_____ К.І. Ліщук

Київ – 2020 року

Структура генеративної мережі



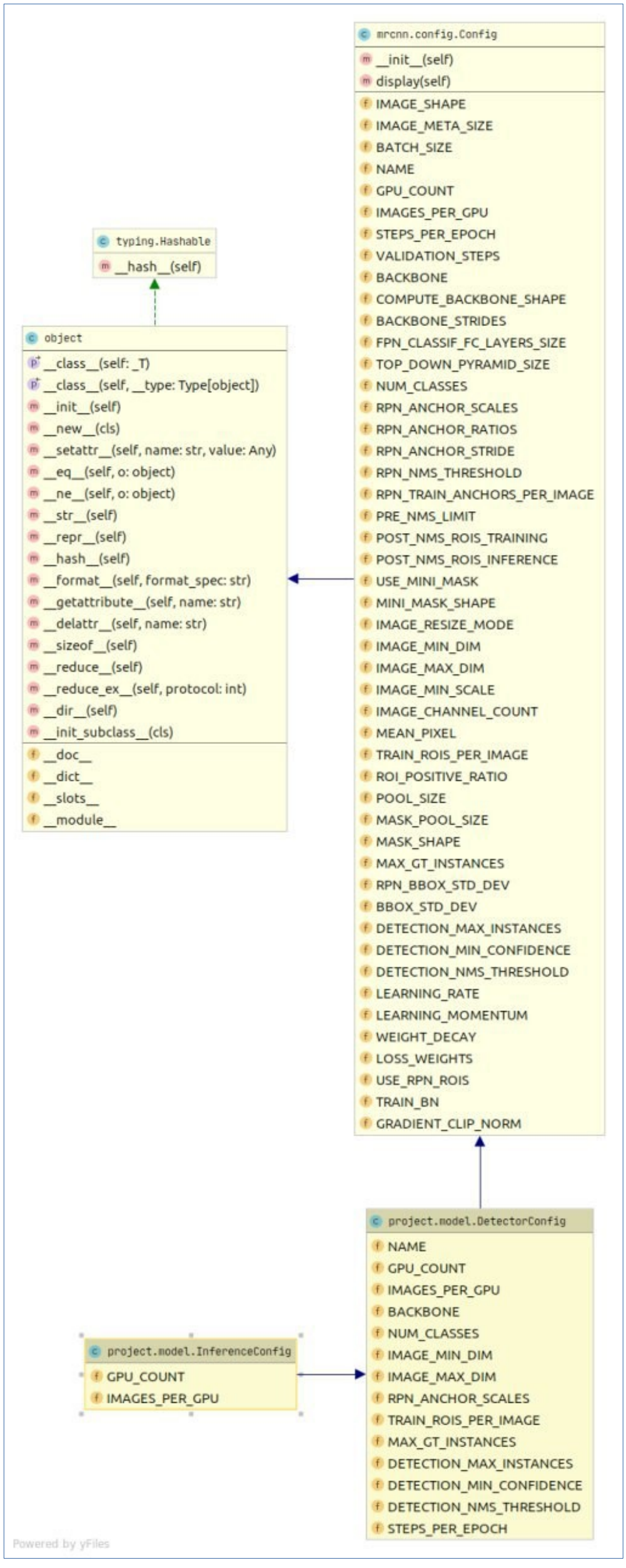
Демонстраційний плакат до дипломного проекту

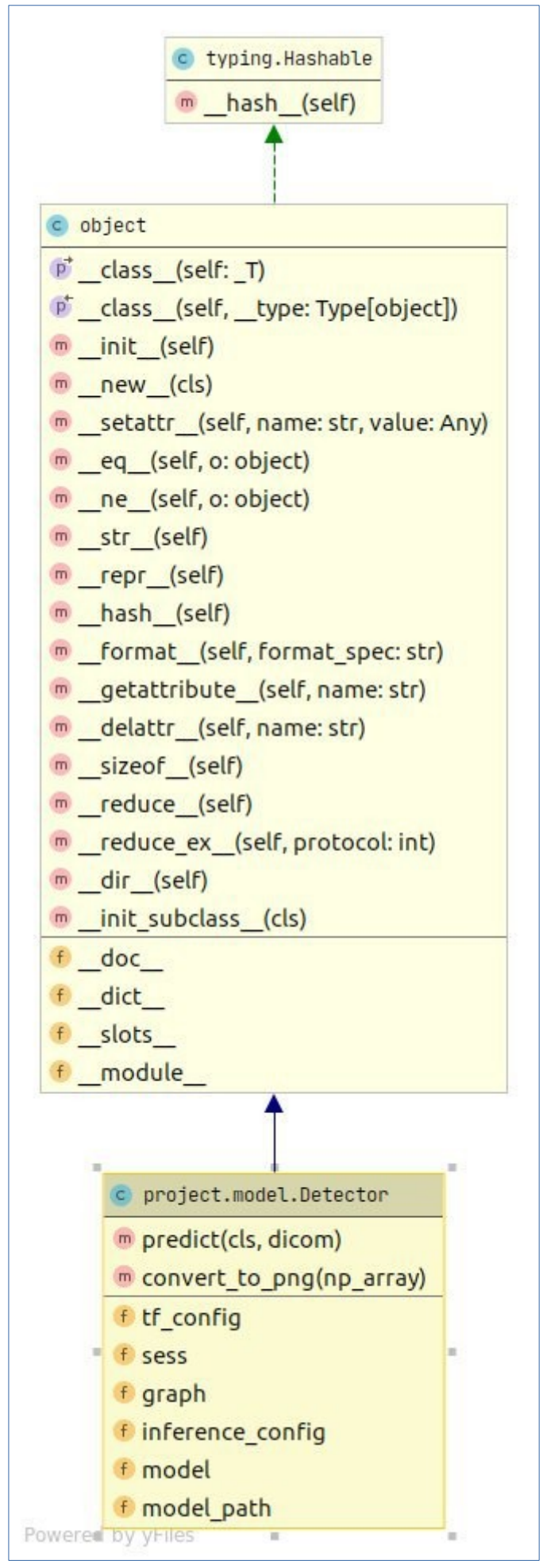
«Структура генеративної мережі»

Виконав студент гр. ІП-61 Данилюк М.І.

Керівник Сирота О. П.

КПІ.ІП-6107.045490.06.99 СС

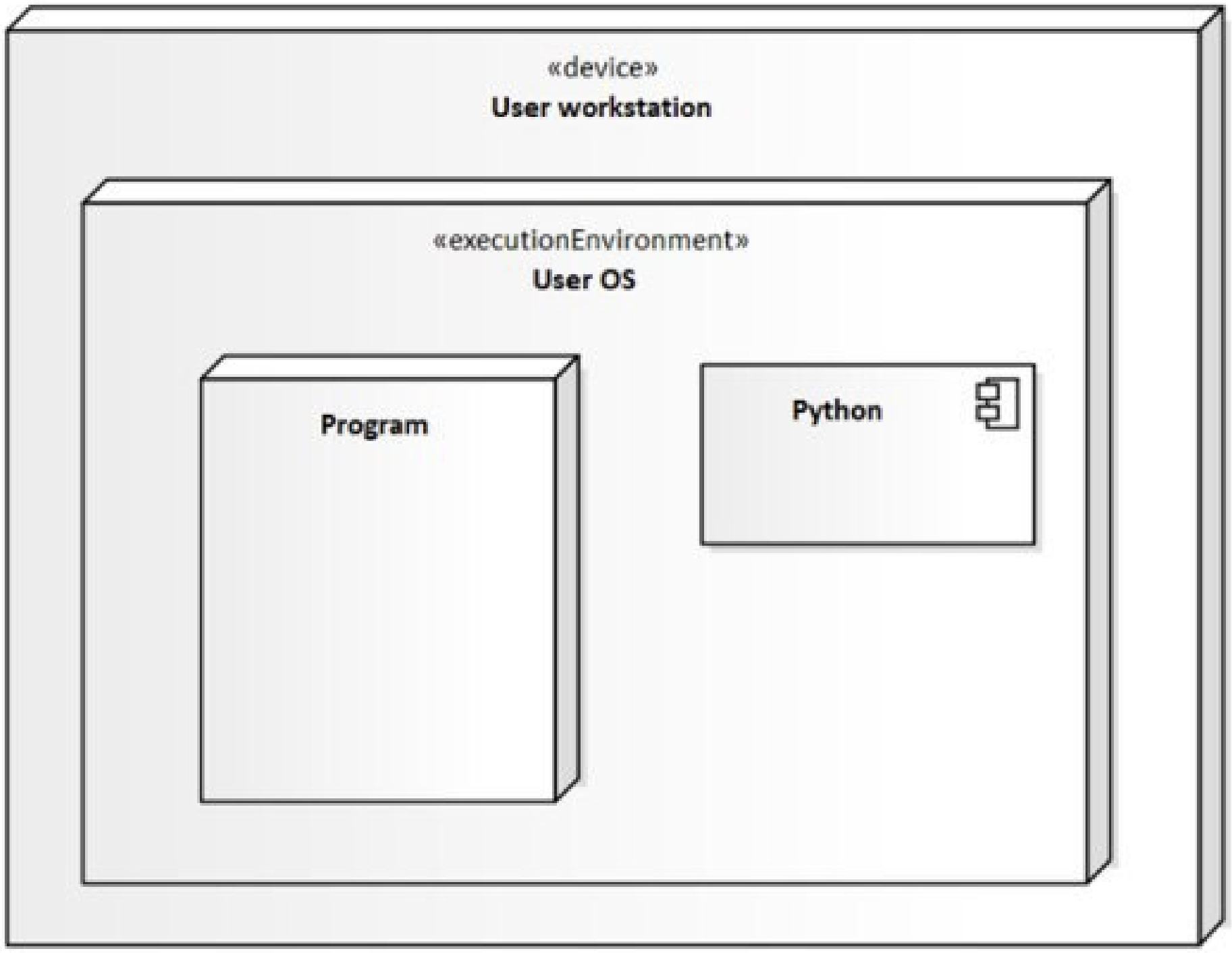




					КПІ.ІП-6107.045490.06.99 СС
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна класів програмного забезпечення
Розроб.		Данилюк М. І.			Лит. / Арк. / Аркушів: 1
Перев.		Сирота О. П.			Аркуш / Аркушів
Т. Кон.					Програмне забезпечення для виявлення затемнених зон на рентген знімку легень
Н. Кон.		Ліщук К.І.			КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-61
Затв.		Сирота О. П.			

Інв. № підп. | Підп. дата | Інв. № взам. підп. | Інв. № дубл. | Підп. дата

КПІ.ІП-6107.045490.06.99 СС



Інв. № підп. | Підп. дата | Інв. № взам. підп. | Інв. № дубл. | Підп. дата

Зм.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-6107.045490.06.99 СС		
Розроб.		Данилюк М. І.			Схема структурна розгортання	Лит.	Арк. / Аркушів: 1
Перев.		Сирота О. П.				Аркуш	Аркушів
Т. Кон.					Програмне забезпечення для виявлення затемнених зон на рентген знімку легень	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-61	
Н. Кон.		Ліщук К.І.					
Затв.		Сирота О. П.					